Gazdaságinformatika
Doktori Iskola

TÉZISGYŰJTEMÉNY

Mohácsi László

Gazdasági alkalmazások párhuzamos architektúrákon

című Ph.D. értekezéséhez

Témavezetők:

Dr. Abaffy József, DSc
Dr. Kovács Erzsébet, CSc

Budapest, 2014

Számítástudományi Tanszék

TÉZISGYŰJTEMÉNY

Mohácsi László

Gazdasági alkalmazások párhuzamos architektúrákon

című Ph.D. értekezéséhez

Témavezetők:

Dr. Abaffy József, DSc
Dr. Kovács Erzsébet, CSc

Tartalomjegyzék

1. Kutatási előzmények és a téma indoklása

Az utóbbi években az egyes számítóegységek műveleti sebességében nem tapasztalható gyors fejlődés. Az alapkutatások terén sem körvonalazódik olyan eredmény, amely az elkövetkezendő években nagyságrendi növekedést hozhatna az általános célú számítóegységek sebességében. Az egy processzorra tervezett programok és algoritmusok futási idejének nagyságrendi javulása nem várható a hardvereszközök fejlődésétől, mert ezek nem tudják kihasználni a rendelkezésre álló további számítóegységeket. Nagyságrendi sebességnövekedés csak a megoldandó feladat részfeladatokra történő bontásával érhető el, melyek megoldása külön számítóegységeken időben párhuzamosan végezhető. Attól függően, hogy a részfeladatok hogyan kapcsolódnak egymáshoz, más-más párhuzamos hardver architektúra nyújt optimális teljesítményt. A több számítási egységből álló architektúrákra történő szoftverfejlesztés egészen más megközelítést igényel, mint a hagyományos, egyprocesszoros architektúrákra történő algoritmustervezés illetve programírás. Gyakran az algoritmus működési elvén is változtatni kell ahhoz, hogy jó teljesítményt nyújtson egy adott párhuzamos architektúrán.

A számítások több számítóegységen történő párhuzamos futtatása nem új találmány. Richard Feynman az atomfegyver kifejlesztését szolgáló Manhattan-terv kapcsán már 1944-ben, még a mai értelemben vett számítógépek megjelenése előtt foglalkozott számítások párhuzamosításával. A feladat a különböző elrendezésű berobbantó bombák energiafelszabadulásának kiszámítása volt IBM gyártmányú program-vezérelt számológépen. Gene Amdahl 1960-ban foglalkozott azzal a problémával, hogy hiába növeljük a párhuzamosan működő adatfeldolgozó egységek számát, a program egy része - melyet az egymásra épülő eredmények miatt nem lehet párhuzamosítani - gátat szab a sebességnövekedésnek.

A piaci nyomás arra ösztönzi a hardvergyártókat, hogy több számítóegységgel rendelkező architektúrákkal jelenjenek meg a piacon, míg az egyes számítóegységek sebességében nem tapasztalható jelentős előrelépés. A különböző architektúráknál a hardverelemek közti kapcsolat jelentősen eltér, így minden architektúra más-más feladattípusoknál nyújt jó teljesítményt. Az algoritmusokat úgy kell fel- illetve átépíteni, hogy illeszkedjenek a futtató architektúrához. A konkrét esetben ideális megoldás megtalálása hardverválasztási és szoftvertervezési kérdéseket vet fel.

Az értekezés három, gazdasági számításoknál és szimulációknál is jelentős algoritmus párhuzamos architektúrára történő újszerű alkalmazásával foglalkozik.

A dolgozat első fejezete leíró, elemző jellegű – a különböző általános célú párhuzamos adatfeldolgozásra alkalmas hardverarchitektúrákról és kapcsolódó szoftverfejlesztő eszközökről nyújt összehasonlító áttekintést. Az itt leírtak alapjául szolgálnak számos későbbi részekben meghozott döntésnek. A fejezet kivonatából készült cikk az NJSzT gondozásában megjelenő GIKOF Journal-ban kerül publikálásra. A témában 2013. novemberében a X. Országos Gazdaság-informatikai Konferencián tartottam előadást.

A második fejezet az ABS lineáris egyenletrendszer-megoldó módszer masszívan párhuzamos architektúrára történő implementációjával és az algoritmus hibaterjedésével foglalkozik. A téma azért aktuális, mert az algoritmus kitűnő stabilitási tulajdonságai 2013-ban kerültek bizonyításra. A problémával mint egy előtanulmánnyal foglalkoztam a masszívan párhuzamos architektúrákra történő algoritmustervezés és implementáció mélyebb megértéséhez.

A harmadik fejezet alapjául Deák Istvánnal közösen írt “A parallel implementation of an $O^*(n^4)$ volume algorithm” című angol nyelvű cikkünk szolgál, amely a Central European Journal of Operations Research-

ben jelent meg. A dolgozatban helyt kaptak azok a magyarázatok is, amelyek a cikkbe terjedelmi okok miatt nem kerülhettek bele. Az elkészült párhuzamos implementáció segítségével az algoritmus viselkedésének tanulmányozására új mélységekben nyílt lehetőség, amelyre korábban sebességkorlátok miatt nem volt mód. Az eredményeket 2013. június 13-án adtam elő a XXX. Magyar Operációkutatási Konferencián.

A negyedik fejezet témájául a nyugdíjrendszer működtetéséhez szükséges modellezéseknél alkalmazásra kerülő előreszámítások egyikét, a demográfiai előreszámításokat választottam. A demográfiai előreszámítások kapcsán kétféle megközelítéssel foglalkoztam – a kohorsz-komponens módszerrel és a mikroszimulációs eljárással. A mikroszimulációs megközelítést mutatom be közelebbről, mivel a nyugdíj előszámításokhoz nem elég a makro szintű megközelítés. A modellezésnél igen fontos a nyugdíjasok és a nyugdíjba vonulók száma mellett azok neme, iskolai végzettsége, a nyugdíjazáskor elért jövedelme, stb. Bemutatom az általam épített mikroszimulációs keretrendszert, működésének szemléltetéséhez a születés és halál előrejelzését dolgoztam ki részletesen. A feldolgozandó rekordok nagy száma és a minden rekordon azonos feladatokat végrehajtó algoritmusok miatt programozás-technikai szempontból a mikroszimuláció jól párhuzamosítható.

2. Felhasznált módszerek

Az első fejezetben közölt állítások nagy részben a mérnökként és szoftverfejlesztőként szerzett sokéves gyakorlati tapasztalaton alapulnak.

A programkódok fejlesztéséhez Visual Studió fejlesztőkörnyezetet használtam CUDA-C illetve beépített C# fordítóval.

Az ABS algoritmus implementációja során a klasszikus vízesés modellt alkalmaztam, melyben először az alap mátrix és vektorműveletek adatmozgatás szempontjából optimalizált változatai készültek el. Ezek helyességét *unit test*ekkel ellenőriztem. A teljes implementáció helyességét vizsgáló modul tesztet véletlen szám generátorral előállított egyenletrendszereken hajtottam végre.

A harmadik fejezet térfogatszámító algoritmusa spirál modell szerint készült – ciklikusan ismétlődtek a fejlesztési fázisok. Az újabb és újabb iterációkra elsősorban a futásidő optimalizálása és az egyre magasabb dimenziókban főként számábrázolásból adódó ritkán előforduló hibák miatt volt szükség.

Véletlen számokra épülő párhuzamos Monte Carlo algoritmusok hibakövetése nehéz feladat mivel párhuzamosan, akár több-ezer szálon zajlanak nem determinisztikus folyamatok. Hiba esetén a hibakövető eszközök a szál korábbi állapotait nem tudják visszavezetni, így nehéz megállapítani, hogyan került a szál hibás állapotba. Az algoritmus helyességének igazolására és viselkedésének mélyebb megértésére vizualizációs eszközöket készítettem, melyek segítségével különböző vetületek mentén vizsgálható a pontok térbeli eloszlása. A moduláris tesztelésre és a nagyszámú egymás utáni futtatás eredményeinek szórásvizsgálatára külön alkalmazás született.

A térfogatszámításnál a párhuzamos implementáció során a Mersenne-Twister véletlen szám generátor NVidia által kiadott implementációjára építettem. Ezt azért fontos megemlíteni, mert mind a térfogatszámításnál, mind pedig a mikroszimulációs keretrendszernél a véletlen számok előállítása teszi ki a futásidő jelentős részét. A khi négyzet és az inverz khi függvények numerikus implementációit az AlgLib függvénykönyvtár szükséges függvényeinek átültetésével készítettem el.

A varianciacsökkentő eljárások fejezetnél a sokdimenziós ortogonális vektorrendszer előállítására először a Gram–Schmidt, később stabilitási okokból a Householder eljárásokat alkalmaztam.

A mikroszimulációs keretrendszer C# nyelven készült – szintén spirál modellt követve. A C# beépített véletlen számgenerátora a Knuth-féle szubsztraktív algoritmusra épül, de nem alkalmas arra, hogy több szálról egyszerre hívják. A probléma megoldására több megközelítést hasonlítottam össze futásidő szempontjából – a klasszikus zárak alkalmazása tűnt optimálisnak. A továbbvezetett állományokból történő lekérdezésre szálbiztos LINQ technológiát használtam.

3. Az értekezés eredményei

Az ABS algoritmussal kapcsolatban megfogalmazott tézisek:

T1. A módosított Huang módszer nagy dimenzióban is megjelenő stabilitása alátámasztja Gáti Attila dolgozatában állított stabilitást.

T2. A módosított Huang algoritmus stabil, a generált p_i vektorok ortogonálisak, amennyiben a $H_1=I$ mátrixszal indítunk. Alacsony memóriaigénye miatt különösen alkalmas GPU-n történő futtatásra.

T3. Parlett és Kahan bebizonyította, hogy az általuk a CGS klasszikus Gram-Scmitt módszerre kidolgozott "twice is enough" eljárás lényegesen pontosítja a lineáris egyenletrendszer megoldását. Ez az eljárás alkalmazható a módosított Huang módszerre is, amely szintén ortogonális vektorokat gyárt, és az általam alkalmazott ABS projekciós mátrixával való újravetítés ezt a funkciót látja el. A megoldás javítása nagy dimenzióban (8000 dimenzióban) telt mátrixokra is működik. Ezt igazolják a teszt feladatok.

Az ABS algoritmus futásidejét 4000 ismeretlenes sűrű együtthatómátrixú egyenletrendszernél GeForce GTX 570 grafikus kártyán 100 másodpercre sikerült leszorítani.

A Lovász-Vempala algoritmussal kapcsolatban megfogalmazott tézis:

T4. Az LVD algoritmus csak egy pontszálat léptetett, ezért pontok függetlenségének biztosításához nem használ fel minden pontot az integráláshoz, hanem egy fázisonként változó alacsony értéknek megfelelő számú pontot minden integrálás után kihagy. A disszertációban bemutatott PLVDM több-ezer pont-szálat léptet. Ebből adódóan nem kell kihagyni pontokat az integrálások között, mert a pontok száma már olyan magas, hogy a kihagyott pont helyén (vagy legalábbis közvetlen közelében) hamarosan úgyis keletkezne egy másik pont. A számítógépes kísérletek alátámasztották azt a feltevést, hogy a több pont-szál használata hozzájárul a generált pontok függetlenségének biztosításához. Ezáltal a Lovász-Vempala algoritmus PLVDM implementációja nem csupán a számítóegységek száma mentén ér el hatékonyságnövekedést. A PLVDM algoritmus által bevezetett varianciacsökkentő eljárások bár javítják az algoritmus hatékonyságát, nem váltották be a hozzájuk fűzött reményeket.

A PVLDM implementáció mellé elkészítettem azokat a segédprogramokat is, melyek lehetővé tették a pontok terjedésének vizualizációját is. A vizualizációnak nemcsak az algoritmus viselkedésének mélyebb megismerésében van jelentősége, hanem fontos szerepe volt a hibák kiszűrésénél is. A dolgozatban bemutatott PLVDM térfogatszámító algoritmus a duplapontosságú számábrázolás hibájából fakadóan 20 dimenziónál korlátba ütközött. A dimenziószám további növeléséhez az algoritmus néhány pontján növelni kell a számábrázolás pontosságát, de ez GPU-n nem oldható

meg hatékonyan. A párhuzamosítás ilyen típusú gépre is megengedi a 20 dimenziós térfogatok elfogadható időn belül történő meghatározását.

A mikroszimulációs keretrendszerrel kapcsolatban megfogalmazott tézisek:

T5. A hagyományos mikroszimulációs szolgáltató rendszerek futásának többórás várakozási idejét a mai felhasználó nem tolerálja – fel kellett gyorsítani a jól ismert algoritmusokat. A párhuzamos processzálási technikával sikerült a futásteljesítményt a végfelhasználók által elfogadható szintre hozni.

T6. Továbbra is cél, hogy a mikroszimulációs keretrendszerek jól paraméterezhetők legyenek, és különböző célú feladatokra legyenek alkalmazhatók. A kidolgozott rendszer paraméterezhető, mind a feldolgozott adatállomány szerkezetét illetően, mind a becslési algoritmusok, és annak paraméterei, mind a mikromodul építés és annak futtatási sorrendjének meghatározása szempontjából.

A KSH-ból származó kutatóállományon végrehajtott 50 éves előrejelzések mikroszimulációs technológiája a párhuzamos programozási technológiával személyi számítógépen 2 percre volt redukálható, mely jó használható a modellező közgazdászok számára. A keretrendszer lekérdező funkciójával létrehozott eredménytáblázatok és korfák igazolják a kidolgozott algoritmus helyességét. A keretrendszer működésének bemutatására a nyugdíjrendszer modellezéséhez szükséges előreszámítások egyikét, a népesség előreszámításokat választottam. Ezek a tényezők – egyebek mellett – döntően befolyásolják a mindenkori potenciális járulékfizetők, a majdani nyugdíj-várományosok számát és a munkaerő kínálati oldalt. A kialakított keretrendszer a nyugdíjrendszerekhez kapcsolódó más tényezők számításaira is alkalmazható. Felhasználható már kialakított modellek továbbfejlesztéséhez, illetve meglévők folyamatos karbantartására, javítására is.

4. Következtetések összegzése

A gazdasági életben hozott döntéseket támogató számítások között sok a nagy számításigényű probléma. A különböző véletlen számokon alapuló Monte Carlo szimulációk, az összefüggő valószínűségek modellezései is nagy számításigényű feladatok, ahol az eredmény pontosságát a végrehajtott szimulációs lépések száma döntően befolyásolja. A gyakorlatban egy számítás elvégzésére rendelkezésre álló idő általában korlátozó tényező.

A piacon elérhető számítóegységek számítókapacitása a fizikai korlátokat közelíti, ebből adódóan az időegység alatt elvégzett műveletek terén nagyságrendi előrelépés csak több számítóegységből álló párhuzamos architektúra alkalmazásával érhető el. A párhuzamos programfuttatásra alkalmas, több számítóegységet tartalmazó eszközök lassan megtalálhatók minden számítógépben és mobiltelefonban. A feladat olyan algoritmusok tervezése, amelyek hatékonyan tudják kihasználni a rendelkezésre álló további számítóegységeket.

Az első lépés a feladathoz legjobban illeszkedő architektúra megtalálása. Sok probléma esetén egyedi fejlesztésű célprocesszor nyújtaná a legjobb teljesítményt, de a költségek és az időkorlátok miatt általában be kell érni valamely általános célú párhuzamos architektúra alkalmazásával. (Az FPGA-k hibrid megoldásként lehetőséget adnak célprocesszorok kialakítására programozottan összekapcsolt logikai kapukból, de a fejlesztéshez szükséges időbeni és anyagi ráfordítások miatt az FPGA alapú megoldások aránylag ritkák.) A különböző architektúrák más-más feladattípusok esetén nyújtanak jó teljesítményt. Általános célú párhuzamos architektúrából többféle áll rendelkezésre a piacon. A dolgozatban szereplő algoritmusok tapasztalatai segítenek a választásban. Az első fejezet áttekintést ad arról, hogy mely általános célú architektúra milyen feladattípusnál hozhat jó teljesítményt. A párhuzamos architektúrákra történő algoritmustervezés és

szoftverfejlesztés egészen más megközelítést igényel, mint az egyszálas megközelítés. A dolgozat írása során is beigazolódott, hogy a meglévő algoritmusok implementációja párhuzamos architektúrán nem csupán szoftverfejlesztési szakmunka, hanem komoly tervezést igénylő matematikai és mérnöki feladat, mely sok intuitív ötletet is igényel. Nem létezik olyan automatizálható módszertan, mellyel egy eredetileg egy processzorra tervezett algoritmus hatékonyan átültethető lenne több párhuzamos rendszerre.

A gazdasági élethez kapcsolódóan sok feladat vezethető vissza lineáris egyenletrendszerek megoldására. A sokismeretlenes egyenletrendszerek megoldása komoly számítókapacitást igényel, valamint számolni kell a gépi számábrázolásból és az algoritmusok instabilitásából adódó hibákkal is. A lineáris egyenletrendszerek megoldása elsősorban mátrix és vektorműveleteket igényel, melyek sajátossága, hogy nagyon sok adatot mozgatnak meg, melyen egyszerű összeadás és szorzás műveleteket kell csak végrehajtani. A GPU architektúrák széles adatbuszuk miatt különösen alkalmasak mátrixművletek felgyorsítására. Az ABS algoritmus módosított Huang változata az adatmozgatások mintázata és a kedvező memóriaigény miatt különösen jól illeszkedik a CUDA architektúrához. Az adatmozgatások optimalizálása után egy 4096 ismeretlenes egyenletrendszer megoldását GeForce GTX 570-es kártyán 100 másodperc környékére sikerült leszorítani, miközben az algoritmus kiváló stabilitási tulajdonságai empirikusan is igazolást nyertek.

Nagyon nehéz jó becslést adni arra, hogy egy adott probléma megoldásánál mely konkrét párhuzamos architektúra ténylegesen milyen teljesítményt nyújt a gyakorlatban. Ez különösen igaz a CUDA architektúrára, ahol a párhuzamosan működő aritmetikai egységek csoportonként közös vezérlőre vannak kötve. Ebből adódóan a közös vezérlőn párhuzamosan futó szálaknak feltételes elágazások és eltérő lépésszámú ciklusok esetén be kell várniuk egymást. A harmadik fejezet írása közben szerzett gyakor-

lati tapasztalat is azt mutatja, hogy az algoritmus fejlesztésének előrehaladtával egyre több eset kerül azonosításra és kezelésre, és emiatt rohamosan nő a feltételes elágazások száma a programban. Ez azzal járhat, hogy a kísérleti projekt alapján becsült, ígéretes futási sebességtől a tényleges futási sebesség fokozatosan elmarad. Ilyenkor az algoritmus működésén kell módosítani ahhoz, hogy egy adott architektúrán jó teljesítményt nyújtson. Jó példa erre dolgozat harmadik fejezetében szereplő Lovász-Vempala térfogatszámító algoritmus, ahol többek között a sokdimenziós testek kezelésén kellett változtatni ahhoz, hogy az algoritmus illeszkedjen a CUDA architektúrához. Párhuzamos programozási technikák alkalmazásával az algoritmus módosított változatán (PLVDM) keresztül először vált vizsgálhatóvá a Lovász-Vempala algoritmus viselkedése magasabb dimenziókban, és a gyakorlatban is alkalmazható implementáció született.

A fejlesztéshez szükséges emberi erőforrást különösen a közösen használt memóriaterületeket zároló, bonyolult, többszálú algoritmusok esetén nehéz becsülni. A zárak feloldására váró szálak között például körbetartozáshoz hasonló helyzet alakulhat ki, melyből a program nem tud elmozdulni. A szálak futási sebessége nem determinisztikus, éppen ezért bizonyos hibák csak ritkán, bizonyos időbeni együttállásoknál jelentkeznek, ezért nagyon nehéz őket tetten érni és reprodukálni. Ez bizonytalan működéshez, illetve időben elhúzódó fejlesztésekhez vezethet. Viszont gondos tervezéssel és implementációval nagy számításigényű problémák is elfogadható időn belül oldhatók meg mindenki számára elérhető általános célú hardveren.

Ilyen például Magyarország népességének mikroszimulációval történő továbbvezetése. A mikroszimulációs módszer egyesével követi a népesség minden egyedének sorsát tipikusan éves szimulációs lépésekben. A negyedik fejezetben bemutatott mikroszimulációs keretrendszer egyfajta „program a programban”. Az egyed sorsáról – elhalálozás, szülés, házasodás, válás, stb. a szimulációs lépésekben történik döntés paraméter-táblák alapján. A szimulációs lépés programkódja, az un. mikromodul aránylag egy-

szerű, a nem informatikus végzettségű elemző közgazdász számára is könnyen értelmezhető, módosítható. A szimulációs lépés kódjának változtatásával, illetve a születési és halálozási valószínűségeket tartalmazó paramétertáblák alakításával sokféle szcenárió és jövőalternatíva kipróbálható. A mikroszimulációs keretrendszer, mely a népszámlálási adatokból kiindulva a teljes népesség adatait képes beolvasni és ezeken a szimulációs lépéseket évenként végrehajtani összetettebb, párhuzamos programozási technikákra épül. A keretrendszer tervezése és felépítése szoftvertechnológiai jellegű feladat volt.

Több architektúrához és szoftverfejlesztő környezethez rendelkezésre állnak olyan monitorozó eszközök, amelyek segítségével meg lehet határozni a program futását korlátozó szűkös erőforrást.

A párhuzamos algoritmusok tervezőjének jól kell ismernie azt az architektúrát, melyre az algoritmust tervezi; az algoritmustervezési és a szoftverfejlesztői munka összefonódik.

5. Főbb hivatkozások

Abaffy J. (1979): A lineáris egyenletrendszerek általános megoldásának egy direkt módszerosztálya. Alkalmazott Matematikai Lapok, 5, 223-240

Abaffy J./Spedicato, E./Broyden, C.G (1984): A Class of Direct Methods for Linear Equations. Numerische Mathematik, 45, 361-376

Abaffy J./Spedicato, E. (1989): ABS projection algorithms: mathematical techniques for linear and nonlinear equations.

Csicsman J. (1987): A mikroszimulációs rendszer számítástechnikai hátterének kialakítása., (KSH) A Háztartási Mikroszimulációs Rendszer munkálatai, Ts-3/8/8 tanulmánysorozat, 1. kötet

Csicsman J. (2001): A BME PIKK Mikroszimulációs projektjének célkitűzései és meggondolásai., (V. Pénzinformatikai Konferencia, Budapesti Műszaki és Gazdaságtudományi Egyetem, 2001. Október 15-16.)

Csicsman J./Fényes C. (2003): A Mikroszimulációs Szolgáltató Rendszer fejlesztése. Alma Mater sorozat - Üzlet, folyamat, monitoring 91

Csicsman J./László A. (2012): Microsimulation Service System. Hungarian Electronic Journal of Sciences

Deák I. (1979): Comparison of methods for generating uniformly distributed random points in and on a hyperspere. Problems of Control and Information Theory, No. 8, 105-113

Deák I. (1990): Random number generators and simulation, in: Mathematical Methods of Operations Research (series editor A. Prékopa). Budapest: Akadémiai Kiadó

Deák I. (2002): Probabilities of simple n-dimensional sets in case of normal distribution. IIE Transactions (Operations Engineering), No. 34, 1-18

Deák I. (2011): E-ciency of Monte Carlo computations in very high dimensional spaces. Central European Journal of Operations Research, 19, 177-189

Dyer, M./Frieze, M./Kannan, R. (1991): A random polynomial-time algorithm for approximating the volume of convex bodies. Journal of the Association for Computing Machinery, 38, 1-017

Gáti A. (2013): Automatic roundoff error analysis of numerical algorithms. Ph.D thesis, Apllied Informatics Doctoral School, Óbuda University

Goetz, B./Peierls, T. (2006): Java Concurrency in Practice. Addison-Wesley 92

Hablicsek, L. (2007): Társadalmi-demográfiai előreszámítások a nyugdíj-rendszer átalakításának modellezéséhez., Jelentés a Nyugdíj és Időskor Kerekasztal számára

Methuen Haque, Imran S/Pande, Vijay S (2010): Hard data on soft errors: A large-scale assessment of real-world error rates in gpgpu. In Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing. IEEE, 691-696

Kannan, R./Lovász L./Simonovits, M. (1997): Random walks and an O*(n5) volume algorithm for convex bodies. Random Structures and Algorithms, 11, 1-50

Kiss T./Csata I. (2007): A magyar népesség előreszámításának lehetőségei Erdélyben. Demográfia No. 7

Klevmarken, N. A./Olovsson, P. (1996): Direct and behavioral effects of income tax changes: simulations with the Swedish model MICROHUS. Amsterdam: Elsevier Science Publishers

Kovács E. (2010): A nyugdíjreform demográfiai korlátai. Hitelintézeti Szemle, 2, 128-149

Lovász L./Deák I. (2012): Computational results of an O*(n4) volume algorithm. European Journal of Operational Research, 216, 152-161

Lovász L./Simonovits M. (1992): On the randomized complexity of volume and diameter. In 33rd IEEE Annual Symp. on Foundations of Comp. Sci., 482-491

Lovász L./Simonovits M. (1993): Random walks in a convex body and an improved volume algorithm. Random Structures and Algorithms, No. 4, 359-412

Lovász L./Vempala, S. (2003): Simulated annealing in convex bodies and an O*(n4) volume algorithm. In Proc. of FOCS., 650-659 93

Lovász L./Vempala, S. (2006): Simulated annealing in convex bodies and an O*(n4) volume algorithm. J. of Computer and System Sciences, 72, 392-417

O'Donoghue, Cathal (2001): Dynamic Microsimulation: A Methodological Survey. Brazilian Electronic Journal of Economics 4 No. 2

Parlett, B.N. (1980): The symmetric Eigenvalue Problem, Englewood Cliffs, N. J. Prentice-Hall

Romeijn, E./Smith, R.L. (1994): Simulated annealing for constrained global optimization. J. of Global Optimization, 5, 101-126

Sanders, J./Kandort, E. (2010): CUDA by example: an introduction to generalpurpose GPU programming. Addison-Wesley Professional , 312 pages

Simonovits M. (2003): How to compute the volume in high dimensions. Math. Programming Ser. B. 97, 337-374

Smith, R.L. (1996): The hit and run sampler: a globally reaching Markov chain sampler for generating arbitrary multivariate distribution. In Proc. 28th Conference on Winter Simulation., 260-264 94

Zafír M. (1987): A háztartási mikroszimuláció. Koncepció, rendszerleírás., (KSH) A Háztartási Mikroszimulációs Rendszer munkálatai, Ts-3/8/8 tanulmánysorozat, 1. kötet

6. A témakörrel kapcsolatos saját publikációk jegyzéke

Csetényi A./ Mohácsi L./ Várallyai L. (2007): Szoftverfejlesztés. HEFOP, Debrecen, ISBN : 978-963-9732-56-8 p. 157

Forgács A. / Mohácsi L. (2014): Gazdasági számítások párhuzamos architektúrákon. GIKOF Journal, HU ISSN 1588-9130, pp. 6-14.

Mohácsi L./Rétallér O. (2013): A mechanical approach of multivariate density function approximation. Proceedings of the International Conference on Modeling and Applied Simulation, ISBN 978-88-97999-17-1, pp. 179-184.

Mohácsi L. / Deák I. (2014): A parallel implementation of an O*(n4) volume algorithm. Central European Journal of Operations Research, DOI :10.1007/s10100-014-0354-7.