MŰEGYETEM 1782

Párhuzamos algoritmusok nemlineáris peremérték-feladatok megoldására

A Budapesti Műszaki és Gazdaságtudományi Egyetem Irányítástechnika és Informatika Tanszékén a PhD fokozat követelményeinek részleges teljesítéseként benyújtott disszertáció rövid összefoglalása

Szeberényi Imre

Tudományos vezető: Dr. Domokos Gábor
Tanszékvezető: Dr. Arató Péter

Budapest, 2003.

Tartalomjegyzék

1. Bevezetés **2**

2. Előzmények **2**

3. Célkitűzés **3**

4. A kutatás eredményei, alkalmazott módszerek **3**

Új tudományos eredmények összefoglalása **11**

Az új tudományos eredmények hasznosítása **11**

Publikációk a disszertáció témakörében **12**

Egyéb publikációk **12**

1. Bevezetés

A dolgozat a mechanikában gyakran felbukkanó, úgynevezett nemlineáris peremérték-feladatok megoldására mutat be új algoritmusokat. Peremérték-feladatként fogalmazhatók meg azok a jelenségek, melyeknél a független változó (idő, ívhossz) több értékéhez rendelt függő változókra adódnak kényszerfeltételek. Legegyszerűbb *dinamikai* példaként egy lövedék célba juttatását említhetjük: úgy kell a kezdeti sebesség-komponenseket megválasztanunk, hogy a rögzített kezdeti pozícióból éppen a kívánt célpozícióba érkezzen a lövedék. A *statikai* feladatok közül a legegyszerűbb peremérték-feladat a rugalmas, végén koncentrált erővel terhelt konzol. A kezdeti (befogási) pontban kell oly módon megválasztanunk a nyomatékot, hogy annak értéke a végpontban (tehát az erő támadáspontjában) zérus legyen.

Mindkét feladat esetében egy közönséges differenciálegyenlet megfelelő trajektóriájának kiválasztásáról van szó. A vonatkozó differenciálegyenletek általában nemlineárisak, és nagyon gyakran csak közelítő módszerekkel integrálhatók. További nehézséget jelent, hogy a peremérték-feladatok igen gyakran valamilyen kontroll-paramétert (pl. teher nagysága) tartalmaznak. A klasszikus („kézi") eljárások a vonatkozó differenciálegyenlet linearizálásából indultak ki, és így a paramétertől lineárisan függő, zárt alakban megadható megoldáshalmazhoz jutottak.

A nemlineáris módszerek megjelenése azt jelentette, hogy a differenciálegyenletet numerikusan már meg tudták oldani. Megkeresték a paraméter zérus értékéhez tartozó, úgynevezett triviális megoldást, majd kis lépésekben növelve a paraméter értékét, megpróbálták követni a megoldásgörbét fokozatosan közelítő, iteratív lépések közbeiktatásával. Ennek az úgynevezett klasszikus növekményes-iteratív útkövetésnek két nagy hátránya van: egyrészt néha nem vagy csak lassan konvergál az iteráció, másrészt a módszer eleve csak egyetlen görbe megtalálására alkalmas, holott tudjuk, hogy nemlineáris rendszerek rendelkezhetnek több megoldással is.

2. Előzmények

A számítástechnikai háttér erősödése a klasszikus útkövető eljárások sebességét is növelte, alapvető hibáit azonban nem szüntethette meg. Ezek a módszerek a „kézi" módszerek általánosított változatának tekinthetők, és az „öröklött" fogyatékosságoktól a nagyobb gépi kapacitás sem mentesítheti őket.

Gyökeresen más megközelítést alkalmazott Domokos és Gáspár, amikor megfogalmazták az ún. Szimplex[1] Algoritmust: bevezették a megoldásként szóba jöhető trajektóriák megfelelő pontjait tartalmazó Globális Reprezentációs Teret (GRT). A GRT lényegében a vonatkozó differenciálegyenlet módosított fázistere, a módosítás a peremfeltételek rendszerétől függ. A GRT véges dimenziós, és topológiailag korrekten ágyazza be a megoldási görbék alkotta halmazt (ez utóbbit a statikában egyensúlyi útnak, a dinamikában bifurkációs diagramnak nevezik).

[1] A szimplex elnevezés a linearizáláskor alkalmazott geometriai felosztásra utal, és nem a lineáris algebrából ismert matematikai módszerre.

A GRT-ben megfogalmazott peremfeltételekkel a peremérték-feladat átfogalmazható egy nemlineáris algebrai egyenletrendszerré, melyben (n-dimenziós GRT esetén) éppen n ismeretlen és $(n-1)$ egyenlet (függvény) található. Az egyenletrendszer megoldása – tipikusan – egyparaméteres, vagyis egy görbe-halmaz. A Szimplex Algoritmus – nevéhez híven – szimplexekre bontja a GRT-t és a szimplexeken belül lineárisan interpolálja az $(n-1)$ függvényt. Ebben az eljárásban nincsen iteratív lépés, és tetszőleges számú, egymással nem összefüggő megoldásgörbe kiszámítására is alkalmas.

3. Célkitűzés

A Szimplex Algoritmus gyökeresen szakított a kézi módszerekkel, ezzel azonban együtt járt az a hátrány is, hogy jelentős számítási kapacitás nélkül még kipróbálni is alig lehetett, komolyabb feladatok pedig reménytelenül hosszú ideig futottak még a leggyorsabb munkaállomásokon is.

Az algoritmus gyakorlati alkalmazása érdekében nagyságrendekkel fel kellett gyorsítani a számításokat. Mivel a processzorok sebessége nem növekszik ilyen ütemben, *egyetlen* reális alternatívaként az algoritmus párhuzamosítása kínálkozott. Ez a disszertáció központi témája. A párhuzamosítás érdekében rendkívül alaposan át kellett tekinteni egyrészt magát az algoritmust, másrészt tekintettel kellett lenni a jelenlegi és jövőbeni hardver lehetőségekre. Részletesen meg kellett ismerni a mechanikai feladatok számítása során felmerülő esetleges nehézségeket illetve kérdéseket, és tekintettel kellett lenni arra is, hogy az algoritmus felhasználói nem rendelkeznek párhuzamos programozói ismeretekkel.

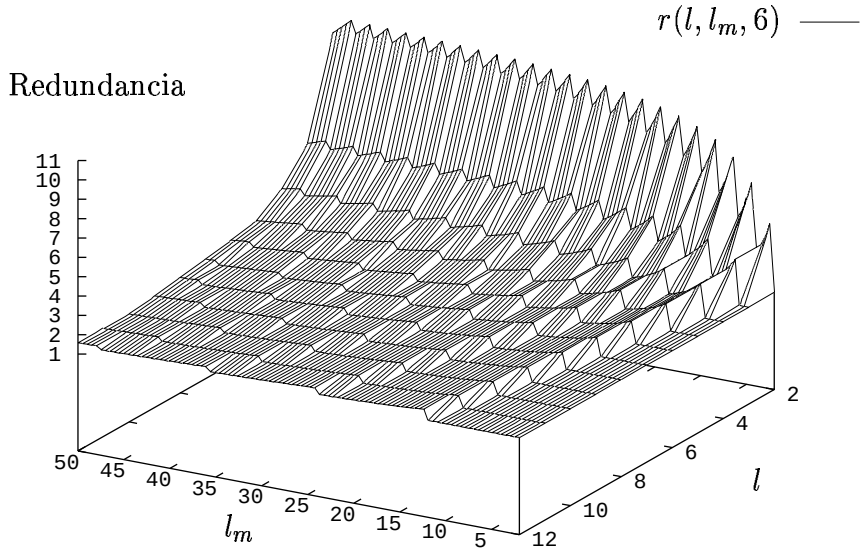
4. A kutatás eredményei, alkalmazott módszerek

Az algoritmus elemzése során elsősorban azt vizsgáltam, hogy mi okozza az algoritmus nagy számítási igényét, és mely részek párhuzamosíthatók a leghatékonyabban. Megállapítottam, hogy alapvetően két tényező a meghatározó:

1. *Szimplektikus rács rácsmérete.*
   A linearizálás miatt csak megfelelően kis méretű szimplexek esetén várhatunk kellően pontos eredményt. Ez azt jelenti, hogy a GRT-t lefedő szimpletikus hálónak nagyon sűrűnek kell lennie, ezért nagyon sok rácspontunk keletkezik.

2. *A numerikus integrátor lépésköze.*
   A számítás során rácspontokban a függvényértékeket minél pontosabban kell előállítani. Miután a függvényértékek a differenciálegyenlet numerikus integrálásával állnak elő, ezért a függvényértékek előállítását végző numerikus integrátornak is kellően kicsi lépésközzel kell dolgoznia.

Az elemzés során megállapítottam továbbá, hogy a gyakorlati esetekben az algoritmus egyéb részeinek hatása elhanyagolható a teljes futási időhöz mérten. Ezt a feltevést kísérleti mé-



1. ábra. Többszörös számításból adódó redundancia (r) a felosztás (l) és a másodlagos rácsméret (l_m) függvényében ($n = 6$). A jobb láthatóság érdekében az ábrát elforgattuk, így az origó a jobb oldalon a távolabbi pontban van.

résekkel[2] is igazoltam. Ezen eredményekre alapozva a további számításokban az algoritmus futási idejét (T) a következő összefüggéssel közelítettem:

$$T = (l+1)^n \cdot k \cdot T_f(n) \tag{1}$$

ahol:

- n - a GRT dimenziószáma
- l - a GRT felosztása (l részre osztunk minden koordinátát)
- k - az integrálási lépésszám
- T_f - a függvények egy pontban történő kiértékeléséhez szükséges idő

Az algoritmus részletes elemzése után számba vettem a lehetséges párhuzamosítási módokat. Ezek előnyeinek és hártányainak mérlegelése során megállapítottam, hogy a GRT tartományaira vonatkozó párhuzamosítási mód a legcélszerűbb megoldás. Ez azt jelenti, hogy a vizsgálandó teret kisebb térrészekre osztjuk, és abban párhuzamosan, egymástól függetlenül végezzük el a számításokat. A feladat ilyen ún. másodlagos felosztásából adódóan a szomszédos térrészek határán levő pontok többszörösen is kiszámítódnak, ami redundanciát eredményez. A számítás redundanciája (r) a számított pontok és a tényleges rácspontok

[2] A méréshez olyan profiler segédprogramot alkalmaztam (gprof), amely a kísérletileg megvalósított algoritmus futási idejének eloszlásáról pontos képet adott.

arányaként képezhető:

$$r(l, l_m, n) = \left(\frac{\left(l + \left\lceil \frac{l}{l_m} \right\rceil\right)}{(l+1)} \right)^n \tag{2}$$

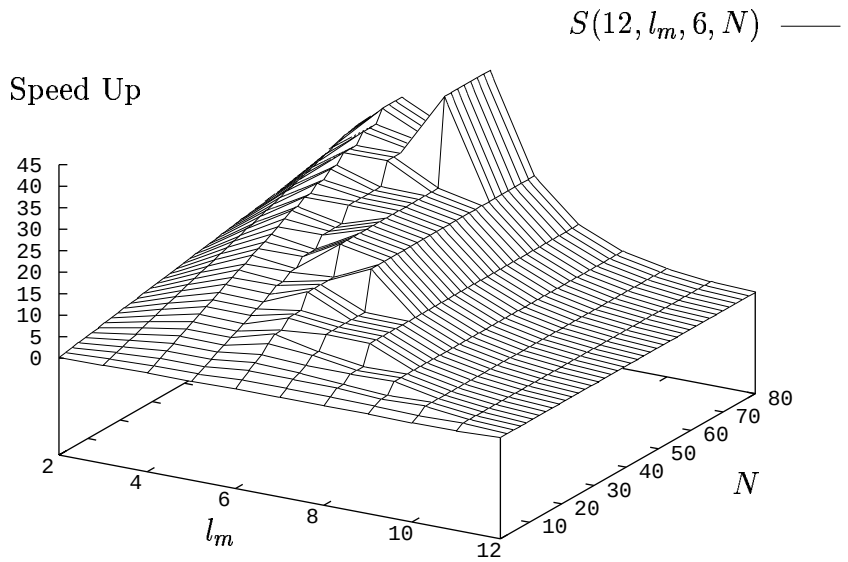
ahol l_m a másodlagos rács rácsméretét jelenti, azaz azt, hogy az l részre osztott koordinátából hány rész esik egy-egy párhuzamosan számított térrészbe. A fenti összefüggés elemzése alapján megállapítottam, hogy hogy kis másodlagos rácsméret esetén kis és nagy felosztás (l) mellett is meredeken megnő a többszörösen számított pontokból adódó redundancia. Nagyobb l_m értékeknél viszont kellően alacsony marad. Ez azt jelenti, hogy az algoritmus megfelelő paraméterezésével a redundancia megfelelő értéken tartható. Az 1. ábra a felosztás és a másodlagos rácsméret függvényében ábrázolja a redundanciát $n = 6$ dimenziós esetben. A 2 összefüggésből az is megállapítható, hogy a dimenziószámtól nem függ a redundancia alakulásának jellege, de kis l_m értékeknél a kifejezés érzékeny a dimenziószám változására.

A választott párhuzamosítási módszer hatékonyságának becsléséhez adatfolyam szinten vizsgáltam a működést. Mester-szolga működést feltételezve egy olyan analitikus modellt dolgoztam ki, amellyel a várható sebességnövekedés (S_N) jól becsülhető:

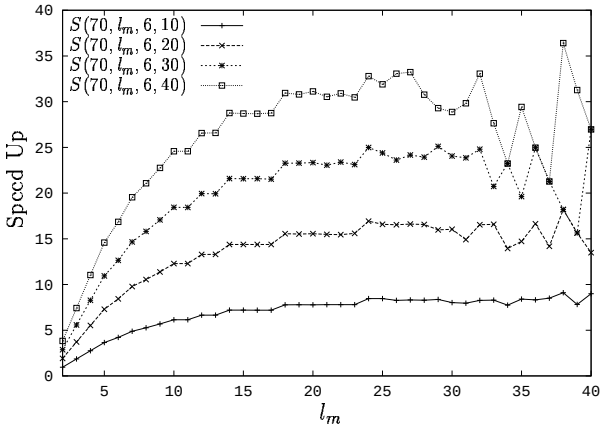
$$S_N = \frac{(l+1)^n}{\left\lceil \frac{\left\lceil \frac{\left(l+\left\lceil \frac{l}{l_m}\right\rceil\right)^n}{(l_m+1)^n} \right\rceil}{N} \right\rceil \cdot (l_m+1)^n} = \frac{1}{\left\lceil \frac{\left\lceil \left(\frac{l+\left\lceil \frac{l}{l_m}\right\rceil}{l_m+1}\right)^n \right\rceil}{N} \right\rceil} \cdot \left(\frac{l+1}{l_m+1}\right)^n \tag{3}$$

ahol N a processzorok számát jelenti. A fenti összefüggés alapján látható, hogy a várható sebességnövekedés görbéje l_m függvényében igen meredeken indul, majd N értékétől függően letörik. A 2. és a 3. ábrák ezt a jelenséget szemléltetik. A 3. ábra a valóságos l/l_m arányokhoz jobban hasonlító esetet mutat, így megfigyelhető a meredek felfutó szakasz. A 4. ábra az N processzorral elért hatékonyságot (E_N) szemlélteti a másodlagos rácsméret és a processzorszám függvényében. Az ábrán jól látható, hogy a felfutó szakaszban a hatékonyság a processzorszámtól gyakorlatilag nem függ.

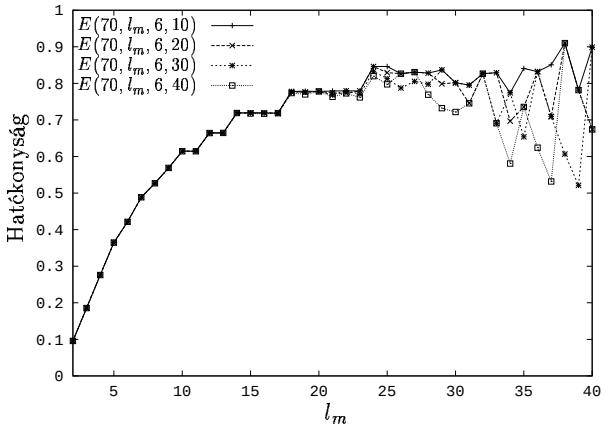
Az analitikus modell alapján elkészítettem az algoritmus mester-szolga modell szerint felépített ún. letapogató változatát. A mester feladata a GRT felosztása, a szolgák közötti feladatok szétosztása és az eredmények begyűjtése. A szolgák feladata a mester által kiszabott térrészre vonatkozó számítások elvégzése és a mesterrel való kommunikáció. A sebességnövekedés előzőekben megadott összefüggését az elkészített párhuzamos program letapogató változatával végzett méréseink igazolták. Az 5. ábra egy ilyen mérési eredmény és a 3 képlet alapján számított értékek összevetését mutatja. A mérést 18 processzorral 6 dimenziós feladaton végeztük. A mérés során viszonylag kis felosztást (12) alkalmaztunk, a másodlagos rács méretét pedig 2 és 12 között változtattuk. Megállapíthatjuk, hogy lényeges eltérést a két görbe nem mutat, kivéve az $l_m = 7$ pontban. Érdekes, hogy a legnagyobb sebességnövekedést nem az elméletileg optimális $l_m = 6$ értéknél, hanem az $l_m = 7$-nél mértük. Ez azzal magyarázható, hogy a mérés folyamán a processzorok teljesítménye ill. terheltsége nem volt azonos. A teljesen szimmetrikus felosztás viszont, mint amilyen az $l_m = 6$, nem teszi lehetővé a gépek más jellegű terheléséből, vagy éppen az eltérő teljesítményükből adódó optimális terheléskiegyenlítést.



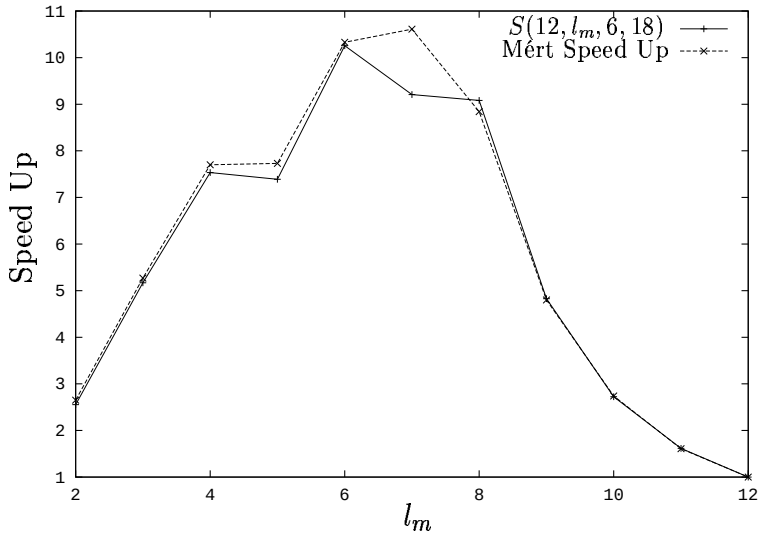
2. ábra. Sebességnövekedés változása a másodlagos rácsméret (l_m) és a processzorszám (N) függvényében ($l = 12$, $n = 6$). A kis felosztás miatt jól megfigyelhető a meredek növekedés, majd a letörés.



3. ábra. Sebességnövekedés a másodlagos rácsméret (l_m) és a processzorszám (N) függvényében ($l = 70$, $n = 6$).



4. ábra. Hatékonyság változása a másodlagos rácsméret (l_m) és a processzorszám (N) függvényében ($l = 70$, $n = 6$).



5. ábra. A 3 összefüggés alapján számított és a program letapogató változatával mért sebességnövekedés változása a másodlagos rácsméret (l_m) függvényében ($l = 12$, $n = 6$, $N = 18$).

A mérés egyéb eredményeit a 4. táblázat tartalmazza. A táblázatban feltüntettük a mért valós futási időket (real), valamint a kommunikációs és az *idle* időket is. Idle időnek azt az időt vettük, amíg a master vagy a slave azért várakozott, mert nem volt mit tennie. A szolgák esetében a mért időket összesítettük. Az eredményekből kitűnik, hogy a kommunikációs idők kis másodlagos rácsméretnél jelentősen megnőnek, de még így is alacsony értéken maradnak. Ez érthető, mivel a mester és a szolgák között átviendő adatmennyiség nem nagy.

Az algoritmus párhuzamos letapogató változatával szerzett tapasztalataink igen kedvezőek voltak, de egyes gyakorlati feladatok megoldásához további teljesítménynövelésre volt szükség. Ez a kutatások folytatására sarkalt. Ennek eredményeként több új algoritmust dolgoztam ki. A kidolgozott algoritmusokat az eredeti algoritmus általánosításaként, közös paraméterezhető algoritmusként fogalmaztam meg. Ezek a következők:

- *Párhuzamos Szimplex Algoritmus (PSA).*
  Ez az eredeti letapagató algoritmus párhuzamos változata, mely a tér teljes letapogatását végzi.

- *Véletlenszerű Szimplex letapogató Algoritmus (VSA).*
  A letapogató algoritmus azon változata, mely a letapogatást valamilyen véletlenszerű sorrendben végzi, és a megtalált megoldásszakaszok számától függően a teljes letapogatás előtt leállítható.

- *Fokozatos sűrítési algoritmus (FSA).*
  A letapogatással nyert eredményeinek pontosítására eredménysen alkalmazható módszer. Alkalmazásával jelentős sebességnövekedés érhető el.

l_m	Pontok	SpUp	Slave idők[s] (szumma)			Master idők[s]		
	(J_m)	S_{18}	real	comm	idle	real	comm	idle
2	34012224	2.65	118344.36			6575.89		
3	16777216	5.27	59470.06	0.669	119.15	3309.76	0.428	3303.24
4	11390625	7.70	40429.51	0.115	331.76	2264.40	0.104	2263.16
5	11390625	7.73	40414.39	0.130	204.72	2256.45	0.181	2255.08
6	7529536	10.33	26946.06	0.014	3438.64	1687.96	0.114	1687.63
7	7529536	10.61	26956.02	0.017	2609.26	1642.44	0.097	1642.13
8	7529536	8.84	26964.22	0.013	8536.06	1972.20	0.054	1971.91
9	7529536	4.80	26915.09	0.020	38435.29	3630.54	0.062	3630.26
10	7529536	2.74	26870.95	0.014	87703.80	6365.24	0.034	6364.99
11	7529536	1.61	27043.06	0.018	167927.12	10831.59	0.099	10831.28
12	4826809	1.00	17451.62	0.005	296678.46	17451.71	0.019	17451.53

1. táblázat. Egy konkrét feladat párhuzamos letapogatásakor mért futási idők és Speed Up (S_{18}) (N=18, l=12). ($l_m = 2$ értéknél a kommunikációs és idle időket mérő alkalmazás (XPVM) hibája miatt nem kaptunk eredményt.)

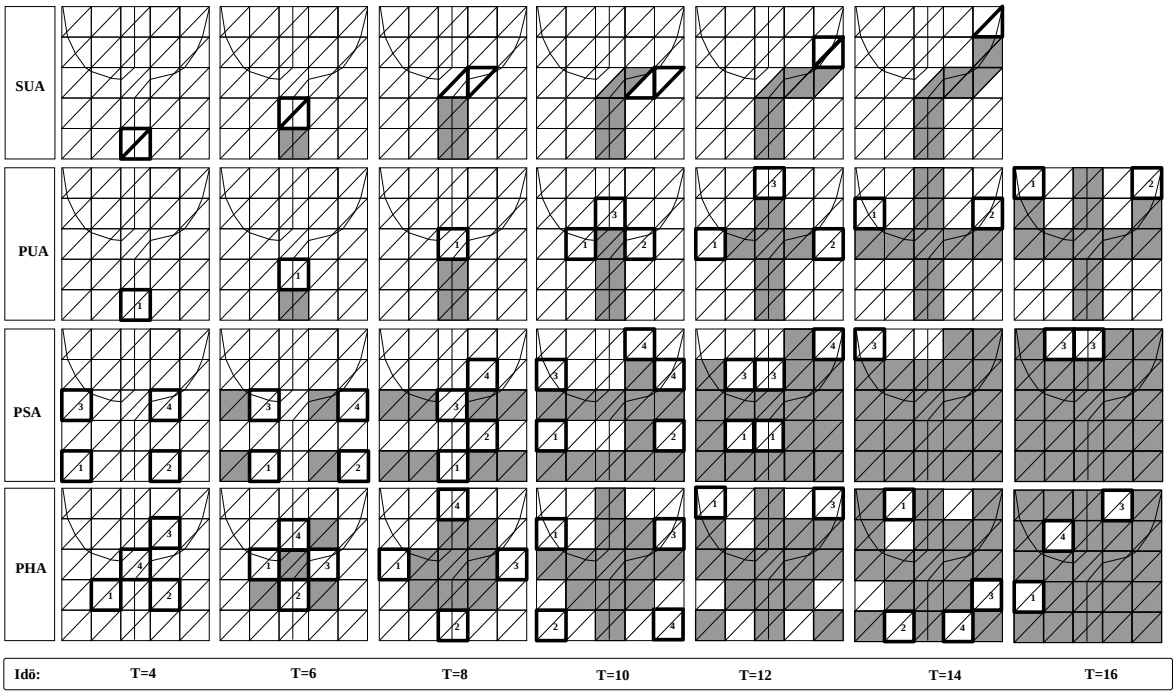
- *Szimplex Útkövető Algoritmus (SUA).*
  A megoldásgörbét szimplexek segítségével követi. (Számos hátrányos tulajdonsága miatt nem implementáltuk.)

- *Párhuzamos Útkövető Algoritmus (PUA).*
  Ez az algoritmus változat a megoldásgörbe több ismert pontjából elindulva párhuzamosan haladva, több ágon követi a megoldást. A hagyományos útkövető módszerekkel ellentétben a követés nem növekményes-iteratív útkövetés, hanem a korábban ismertetett Szimplex Algoritmus útkövető változata, mely mentes az iterációktól. Ez változat képes kezelni az elágazásokat, de a nem összefüggő szakaszokat természetesen nem találja meg.

- *Párhuzamos Hibrid Algoritmus (PHA).* Ez a változat az útkövetést és letapogatást dinamikusan kombinálja. Így ötvözi a letapogató és útkövető változatok előnyeit. Érdekessége a hibrid algoritmusnak, hogy a követés ill. letapogatás „aránya" paraméterezhető, így az algoritmus tulajdonságai széles határok között változtathatók.

A fenti algoritmus-változatok 4 processzoros működését illusztrálja a 6. ábra, amely a bevezetőben említett legegyszerűbb példán, a nyomóerővel terhelt rugalmas konzol egyensúlyi útjának számításán keresztül mutatja be a működést. Az ábrán jól megfigyelhető, hogy az egyes fázisokban mely megoldásszakaszokat találják meg az egyes algoritmusok. Az egyszerűség kedvéért időegységnek egy függvényérték kiszámítását vettük. Megfigyelhetjük, hogy az adott feladattal a PHA algoritmus végez a leghamarabb, 12 időegység alatt. Természetesen az algoritmusok igazi próbáját a valós méretű feladatok jelentik, ezeket azonban nem lehet hasonló módon szemléltetni.

Az eredeti algoritmust nem csak a teljesítménynövelés érdekében fejlesztettem tovább, hanem kifejlesztettem egy olyan változatot is, amely többparaméteres peremérték-feladatok



6. ábra. A különböző módszerek működésének grafikus összehasonlítása

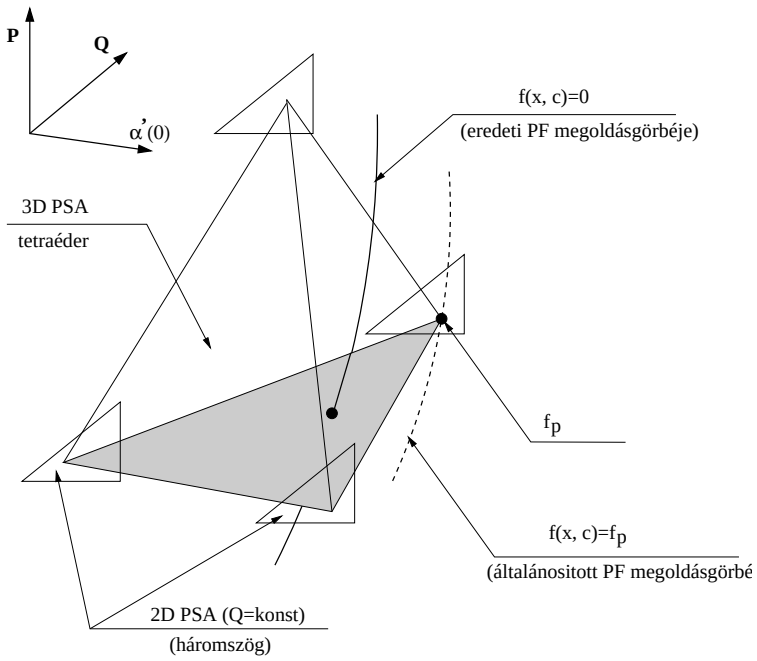
vizsgálatára is alkalmas. Ezt az algoritmus rekurzív kiterjesztésével értem el. Ennek lényege, hogy a problémát egy magasabb dimenziószámú feladatra vezettem vissza, amit rekurzív módon oldok meg. A 7. ábra egy 3 dimenziós feladat rekurzív megoldását szemlélteti.

Az algoritmusokat egységes keretbe foglaló program alkalmazása során szerzett tapasztalatok felhasználásával egy olyan környezetet definiáltam, amely hatékonyan képes segíteni a peremérték-feladatok globális vizsgálatához kapcsolódó kutatói munkát. Meghatároztuk továbbá azokat a tevékenységeket, melyek számítógépes segítésével egy kényelmes környezet alakítható ki. Ezek három fő csoportba sorolhatók:

- *Számítást segítő keretrendszer, segédprogramok.*
  A számítások elvégzéséhez, azok előkészítéséhez és az eredmények előfeldolgozásához egy egységes keretrendszert definiáltam, mely segédprogramjai elfedik a konkrét számítási környezet részleteit. A keretrendszer kialakításához olyan könnyen feldolgozható szöveges állományokat specifikáltam, melyek öndokumentálóak, így hordozzák az ismételt előállításukhoz szükséges összes információt.

- *Eredmények interaktív megjelenítése.*
  Az eredmények kiértékeléséhez olyan grafikus rendszert definiáltam amellyel nem csak az egyensúlyi út, hanem annak tetszőlegesen kiválasztott pontjához tartozó, a problémára jellemző fizikai alak is megjeleníthető.

- *Kutatói együttműködés támogatása.*
  A hatékony kutatói munka támogatására olyan környezetet definiáltam, amely lehetővé



7. ábra. Rekurzió: 2 dimenziós feladat megoldása a 3 dimenziós szimplex csúcsain

teszi egymástól földrajzilag távol levő kutatók konkurens, interaktív együttműködését. Ennek lényege, hogy a képernyőn megjelenő információk mindkét helyszínen azonos időben és módon látszanak.

An n-dimenziós ponthalmazként keletkező eredmények grafikus megjelenítéséhez legeredményesebben az IBM Data Explorer elnevezésű termékét használtam. A DX rendszer alkalmazásával szerzett tapasztalatok alapján, a rendszerhez egy olyan programmodult definiáltam, amely a továbbiakban egyszerűen felhasználható a feladatspecifikus programrészek automatizált elkészítésére.

A kutatói együttműködés támogatására leghasznosabbnak a VNC (Virtual Network Computing) programcsomag bizonyult, amely támogatja a többfelhasználós, konkurens megjelenítést is. Ennek használata azért vált lehetővé, mert a programrendszer megvalósítás során nyílt rendszerkomponenseket használtam. Így a teljes rendszer, beleértve a megjelenítést is egyszerűen összeintegrálható más nyílt rendszerrel. A VNC lehetővé teszi távolsági hálózaton keresztül is a közös interaktív munkát. Ez azt jelenti, hogy azonos képernyőképet láthatunk akár több ezer Km-re levő munkaállomásokon. A rendszer alkalmazásához a kliens oldalon nem szükséges speciális programot telepíteni. Egy egyszerű böngésző is elegendő használatához. Így a számítások grafikus eredménye szinte bárhonnan megtekinthető pl. egy NoteBook-ról, egy PDA-ról, vagy akár egy repülőtéri váróterem nyilvános böngészőjéről.

Új tudományos eredmények összefoglalása

1. Tézis. *A Szimplex Módszer letapogató változatához (PSA) – a párhuzamosítási lehetőségek feltárása és számszerű egybevetése alapján – párhuzamos algoritmust dolgoztam ki és implementáltam. Az elkészített programot heterogén elosztott környezetben és párhuzamos architektúrán is eredményesen teszteltem.*

2. Tézis. *Analitikus közelítő modellt dolgoztam ki a párhuzamosítás hatékonyságának vizsgálatára. A modell érvényességét mérésekkel igazoltam. A kidolgozott modell lehetővé teszi, hogy a letapogató módszerhez a konkrét feladattól függetlenül előzetesen becsülni lehessen a leghatékonyabb feladatfelosztást.*

3. Tézis. *A Szimplex Módszer letapogató változatához (PSA) gyorsító algoritmust készítettem, amely az n-dimenziós térrács lokális sűrítésén alapszik (FSA). Valós és teszt-feladatokon végzett mérésekkel igazoltam, hogy az eredeti, gyorsítás nélküli módszerhez képest – a feladat jellegétől és dimenziószámától függően – 2-150-szeres sebességnövekedés érhető el.*

4. Tézis. *Kidolgoztam a Szimplex Módszer párhuzamos útkövető (PUA), véletlenszerű letapogató (VSA) és hibrid (HSA) változatait. Ezeket a korábban kidolgozott PSA általánosításaként fogalmaztam meg, és közös, paraméterezhető algoritmusként implementáltam. A módszert úgy egészítettem ki, hogy az szakadásos függvényekre is alkalmazható legyen.*

5. Tézis. *Kidolgoztam a Szimplex Módszer általánosított, többparaméteres feladatok megoldására is alkalmas változatát (ilyen feladat pl. a stabilitási határgörbék számítása), melyet egy magasabb dimenziószámú feladat megoldására vezettem vissza. Az új, magasabb dimenziószámú feladat megoldását az általánosított algoritmus rekurzív kiterjesztéseként fogalmaztam meg.*

6. Tézis. *Olyan alkalmazási modellt és környezetet dolgoztam ki, amely a függvénymodul és a konfigurációs állomány megadása után többlépcsős számítást, illetve elosztott hálózati környezetben több felhasználóval együttműködő interaktív megjelenítést tesz lehetővé.*

Az új tudományos eredmények hasznosítása

A kutatómunka eredményei alapján kifejlesztett programrendszer és alkalmazási modell eredményességét mutatja, hogy a rendszert az elmúlt időszakban számos probléma megoldására sikeresen alkalmaztuk különböző kutatási területek kutatóival együttműködve. Ezek $2 \leq n \leq 9$ dimenziós problémákhoz vezettek. A teljesség igénye nélkül felsorolunk ezek közül néhányat:

- gátolt érintkezési feladatok
- folyadékhidak egyensúlya

- mikroszálak konfigurációi
- DNS molekulák mechanikai modellje
- növényi indák mechanikai modellezése

A 8. ábra növényi indák vizsgálatánál kapott eredményeket mutatja, mely probléma 6 dimenziós feladathoz vezetett. Az ábra jobb oldalán látszik a PHA algoritmussal kiszámított 6 dimenziós ponthalmaz egy vetülete, a bal oldalon pedig a ponthalmazból kiválasztott indaalak képének vetületei ill. perspektivikus képe.

Az alkalmazások során mintegy 18 publikáció született közel ugyanennyi szerző tollából. Az algoritmusokat a PVM (Parallel Virtual Machine) rendszer alatt implementáltam, ami lehetővé tette, hogy homogén (cluster, szuperszámítógép) illetve heterogén környezetben is használhatóak legyenek. A párhuzamosítás hatékonyságára jellemző, hogy 120 processzorig a teljes tartományban közel lineáris sebességnövekedést tapasztaltunk.

Publikációk a disszertáció témakörében

[1] G. Domokos, I. Szeberényi, and P. Steen. Parallel, recursive computation of global stability charts for liquid bridges. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface, Proc. 7th EuroPVM/MPI2000*, pages 64–71. Springer (Lec. Notes in Comp. Sci., Vol 1908), 2000.

[2] G. Domokos, I. Szeberényi, and P. Steen. Simulatenously resolved bifurcation diagrams: a novel global approach to liquid figures of equilibrium. *J. Comp. Phys.*, 159:38–57, 2000.

[3] Zs. Gáspár, G. Domokos, and I. Szeberényi. A parallel algorithm for the global computation of elastic bar structures. *Comp. Assist. Mech. and Eng. Sci.*, 4:55–68, 1997.

[4] P. Holmes, G. Domokos, J. Schmitt, and I. Szeberényi. Constrained Euler buckling: An interplay of computation and analysis. *Comp. Meth. in Appl. Mech. and Eng.*, 170 (3–4):175–207, 1999.

[5] Szeberényi I. Elosztott párhuzamos programozás megvalósítása PVM rendszerrel. In *HUNIX '93 konferencia kiadvány*, pages 61–68, Budapest, 1993. NJSZT.

[6] I. Szeberényi and G. Domokos. Solving generalized boundary value problems with distributed computing and recursive programming. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface, Proc. 6th EuroPVM/MPI1999*, pages 267–274. Springer (Lec. Notes in Comp. Sci., Vol 1697), 1999.

[7] I. Szeberényi, G. Domokos, and Zs. Gáspár. Solving structural mechanics problems by PVM and the simplex algorithm. In *Proc. 2nd EuroPVM'95*, pages 247–252. Hermes, Paris, 1995.

[8] I. Szeberényi. PVM implementation of boundary value problems on heterogeneous parallel systems. *Periodica Polytechnica Electrical Engineering*, 44 (3-4), 2000.

Egyéb publikációk

[1] Benyó B., Fék M., Kiss I., Kóczy A., Kondorosi K., Mészáros T., Román Gy., Szeberényi I., and Sziray J. *Operációs rendszerek mérnöki megközelítésben*, pages 374–383. Panem, 2000.

[2] J. Grantner, I. Szeberényi, J. Harangozó, I. Horváth, Cz. J. Horváth, and D. Ujj. The PC-188-S16 I/O interface board. In *Sixth Symposium on Microcomputer and Microprocessor Applications. The Professional Personal Computer and its World*, pages 309–321. HTE, 1989.

[3] Szeberényi I. Virtuális számítógép TCP/IP hálózaton. In *NetworkShop '95 konferencia*, Gödöllő, 1995.

[4] Szeberényi I. Bevezetés a UNIX operációsrendszerbe. Sokszorosított oktatási segédlet 120 o., 1998.

[5] Szeberényi I. Virtuális számítógép a hálózaton. In *NetworkShop '98 konferencia*, Győr, 1998.

[6] Szeberényi I. Nagy feldolgozási képességű számítási környezet a BME-n. In *NetworkShop '2001 konferencia*, Sopron, 2001.

[7] Szeberényi I. and Mohácsi J. TCP/IP-PROFIBUS átjáróprogram megvalósítása. In *microCAD '94 Nemzetközi Számítástechnikai Konferencia*, 1994.

[8] Grantner J., Szeberényi I., Harangozó J., Horváth I., Horváth Cz. J., and Ujj D. PC-188-S16 soros be/kiviteli interfész kártya és magasraktári alkalmazásra. *Mérés és Automatika*, 37(6):365–370, 1989.

[9] Hanák P. and Szeberényi I. Bevezetés az E-levelezésbe. In *Esélyek az ezredfordulón konferencia kiadvány*, Budapest, jún. 21–21 1993.

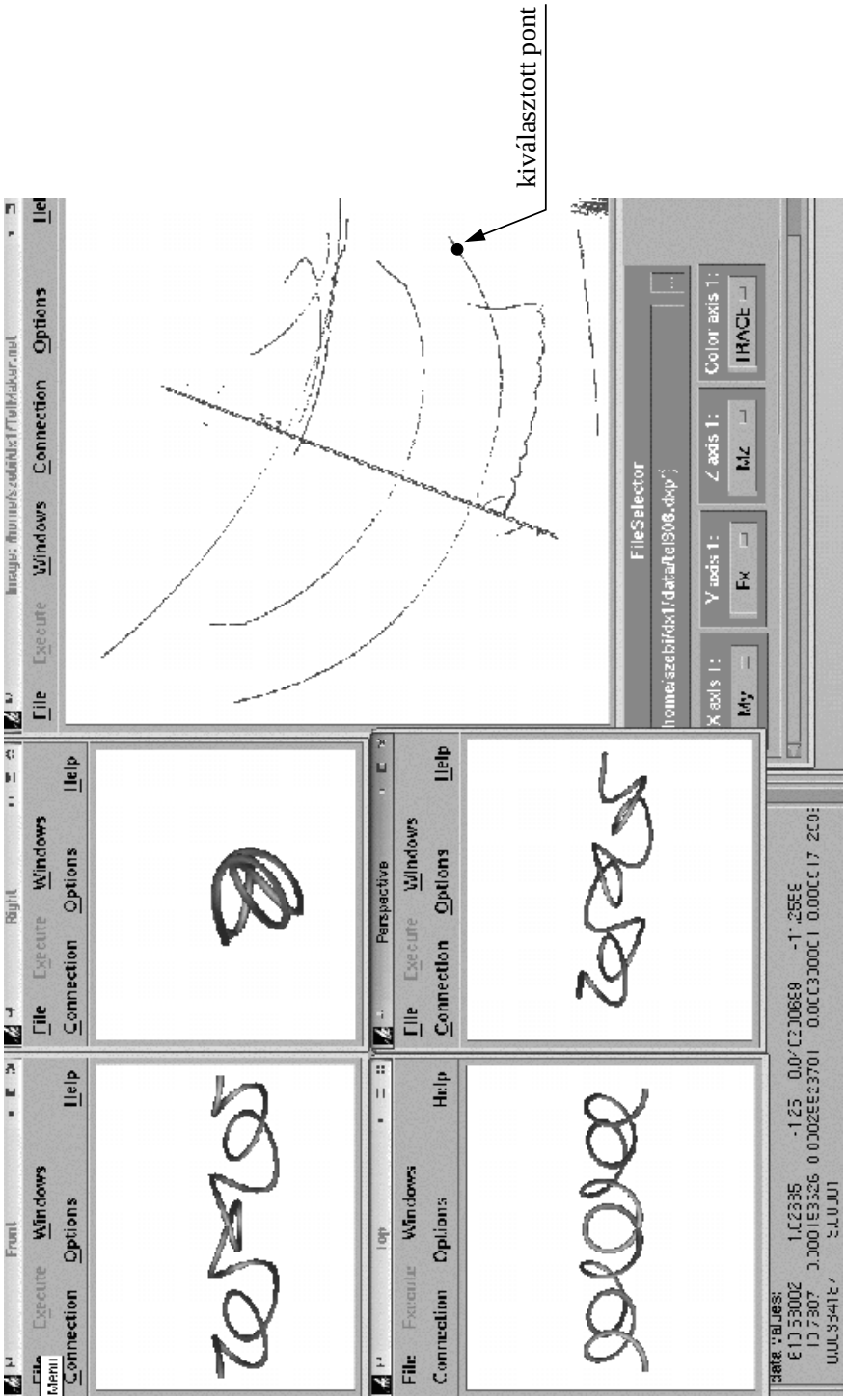
[10] Szigeti Sz. and Szeberényi I. Parallel Virtual Machine programozása. Sokszorosított oktatási segédlet 27 o., 1998.

[11] I. Szeberényi. Applications of cluster computing technology at the TUB. In *US-Hungarian Workshop*, University of Wisconsin, 2001.

[12] Máray T. and Szeberényi I. Unix történelem dióhéjban. A nyíltság nyitja. *Heti Chip*, I.(2):11–17, 1991.

[13] Máray T. and Szeberényi I. UNIX-szal az oktatásban a BME Villamosmérnöki Karán. In *HUNIX '92 konferencia kiadvány*, pages 45–49. NJSZT, 1992.

[14] Máray T. and Szeberényi I. WWW az új multimédia információs technológia. In *DAT '94 konferencia kiadvány*, Budapest, nov 1994.



8. ábra. Növényi indák vizsgálata során keletkezett bifurkációs görbe és a görbe egy pontjához tartozó alak megjelenítése a DX rendszerrel