Prímkeresés villogó áramkörökkel

Pál Ferenc

Babeş-Bolyai Tudományegyetem, Fizika Kar, orvosi fizika szak, 3. évfolyam

Témavezető:

Dr. Néda Zoltán egyetemi professzor

Babeş-Bolyai Tudományegyetem, Elméleti és számítógépes fizika tanszék

XII. Erdélyi Tudományos Diákköri Konferencia

Kolozsvár, 2009. május 15–17.

Kivonat

A számítástechnika fejlődése nemkonvencionális számítástechnikai módszerek kifejlesztését igényli. Az exponenciális számítási sebességnövekedés fenntartása, a növekvő hődisszipáció kezelése, a mesterséges intelligencia kifejlesztése és a gyakorlati szempontból fontos NP-teljes és NP-nehéz feladatok kezelése sok esetben lehetetlen a klasszikus bináris architektúrára épülő számítási paradigmáinkkal. A már-már klasszikusnak nevezhető kvantum-, DNS- és CNN számítógép ötletek mellet, a számítástechnika számos más új úttal is próbálkozik. A jövő a párhuzamos multi-processzálás irányában van. Ígéretes lehetőségeket rejtenek a sok-részecske rendszerekben megfigyelt kollektív viselkedések. A számítástechnika jövője így bizonyos mértékben a fizikusok, biológusok és kémikusok kezében is van. Ezen dolgozat kereteben egy villogó áramkör-sokaságban levő kollektív viselkedést használunk fel prímszámok keresésére. A rendszer és az algoritmus egyszerűsége didaktikusan szemlélteti azon számítástechnikai lehetőségeket amelyek az egyszerű és kölcsönható rendszerek kollektív viselkedésében rejlik.

Bevezető

Az információ* az az absztrakt nyersanyag, amiből a fejlődés építkezik. Az információt viszont elő kell állítani a meglévő adatokból, mozgatni kell, hogy eljusson a megfelelő címzetthez. A nagy méretű adathalmazokat rendezni, szűrni kell, hogy a nekünk fontos, minket érdeklő adatokat kapjuk. A felsoroltak pedig mind-mind a számítógépek feladatai. A számítógép dolgozza fel számításokkal az adatokat és nyújt információt a felhasználónak. A számítógép rendezi és szűri a nagy adathalmazokat, hogy a kívánt információt kinyerhessük belőlük. Manapság az élet minden területén szinte nélkülözhetetlenek a gyors számítógépek.

Egy információ annál hasznosabb, minél pontosabb és minél gyorsabban juthatunk hozzá. Ez a tény volt az, ami meghatározta a számítógépek fejlődését. Ahhoz, hogy az információkhoz oly erősen kötődő fejlődés töretlen maradjon, teljesülnie kell a Moore törvénynek.

A Moore-törvény kimondja, hogy az integrált áramkörökben lévő tranzisztorok száma – ami használható a számítási teljesítmény durva mérésére – minden 18. hónapban megduplázódik.[1] Ez mindezidáig teljesült is, és a számítógépek egységárra vett számítási kapacitása exponenciálisan növekedett. Azonban a jelenlegi tendenciát követve a közeli

* Az információ számunkra eddig ismeretlen, új adatot, ismeretet jelent.

jövőben a miniatürizálással már az atomi méretekig kellene lejutni. Ami viszont azt jelentené, hogy eltűnnének az anyagnak azon tulajdonságai, amiket a jelenlegi félvezető technika felhasznál. Látható tehát, hogy az eddigi út, azaz a miniatürizálás, már sokáig nem követhető. Paradigmaváltásra van szükség, hogy a Moore-törvény továbbra is megmaradjon.

A paradigmaváltás szükségességét a számítógépekkel szembeni újabb igények is jelzik. Például a processzorok energiafogyasztásának csökkentése. Egyrészt az egyre népszerűbb mobil eszközök autonómia-ideje úgy növelhető, ha fogyasztásuk alacsony, másrészt a világon jelenleg létező körülbelül milliárd számítógép összfogyasztása igencsak jelentősnek tekinthető.

A kis méretre csökkentett és nagy frekvenciákon működő processzorok egyre nehezebben hűthetők, mivel kis felületen nehezebb megvalósítani a kívánt hőleadási sebességet.

Megint másrészt a ma használatos számítógépek elvi felépítése, ami a Neumann-elveket követi, nem teszi lehetővé az igazán nehéz feladatok megoldását elfogadható időkorlátok mellett. A Neumann-elvek szerint működő számítógépek processzora a műveleteket meghatározott, egymás utáni sorrendben végzi el. Noha a processzorok órajele GHz nagyságrendű, ami milliónyi elemi művelet elvégzését jelenti másodpercenként, vannak olyan feladatok (NP-teljes vagy NP-nehéz feladatok), amelyekhez ez is kevésnek bizonyul.

A Neumann-elvű számítógépek másik hátránya, bizonyos szempontból, hogy az adatokat csak diszkrét értékekként tudják kezelni és műveleteket is csak diszkrét értékekkel tudnak végezni. Ez azonban a folytonosságot feltételező feladatok esetén, mint amilyenek a parciális differenciál egyenletek megoldása, vagy a numerikus integrálás, nagyon előnytelen. Műtermékek, azaz a valóságnak nem megfelelő eredmények, származhatnak az ilyen számításokból.

A mesterséges intelligencia megvalósítása sem valószínű hogy digitális architektúrájú, a sorrendiség által „akadályozott”, diszkrét időlépésekben működő számítógépen fog megtörténni. A mesterséges intelligenciának adaptívnak, bizonyos mértékben intuitívnak, sőt kreatívnak kell lennie. Ezek a jelzők meg nem igazán a Neumann-elvű számítógépeket írják le. Valamely más elveken alapuló számítógép lesz az amely ezt teljesítheti.

A fentebb felsorolt érvek alapján elmondható, hogy a számítástechnikában paradigmaváltásra van szükség, hogy az eddigi fejlődési ütemet fenntarthassuk, az újonnan támasztott igényeknek eleget tegyenek a számítógépek, az eddig megoldhatatlan és a gyakorlatban fontos feladatokat tudjunk megoldani, valamint mesterséges intelligenciát legyünk képesek létrehozni.

A számítógépek fejlődése

A számítástechnika hajnalát a mechanikus számítógépek jelentették. Az első sikeres összeadógép a mai kilométer-számlálókhoz hasonlóan működött, 0-tól 9-ig számozott fogú fogaskerekekkel. Minden helyiértéknek megfelelt egy számozott fogaskerék. A kerekek úgy kapcsolódtak össze, hogy számokat lehetett összeadni vagy kivonni a fogaskerekek megfelelő számú foggal történő elforgatásával. A gépben működött a tízesátvitel is.[2]

A szorzást egy un. bordás hengerrel oldották meg. A henger felületén 9 db, eltérő hosszúságú borda volt, ezek széles fogaskerék-fogként működnek. A hengerhez illeszkedő számláló fogaskerék saját tengelye mentén eltolható, és megfelelő beállításával elérhető, hogy a bordás henger egy teljes körülfordulása során fogaiba pontosan 1, 2, ... 9 számú borda akadjon be és így ennyi foggal forduljon el a fogaskerék. Ha tehát a fogaskerék tengely menti eltolásával beállítják a szorzandót (hogy hány borda akadjon a fogakba), akkor a bordás hengert annyiszor körbeforgatva, amennyi a szorzó, a fogaskerék a két szám szorzatának megfelelő számú foggal fordul el.[3]

A fent említett két alapvető összetevő (a fogaskerekes tízes-átvitel és a bordás hengerrel történő szorzás) képezte a mechanikus számológépek lényegi egységeit. Ezen két alkotóelem kombinálásával egyre bonyolultabb gépezeteket alkottak meg, amelyek több és több számjegyet tudtak kezelni, és komplexebb számításokat elvégezni. Az összetettebb gépek vezérlését lyukkártyákkal oldották meg. A mechanikus számológépek nyilvánvaló hátrányai, mint a súrlódás, robusztusság, az egyre több mozgó alkatrész, mind-mind a számolási sebesség kárára voltak. Magától értetődően a számítógépnek új területből merítve, új elvek alapján kellett továbbfejlődnie.

A mechanikus számológépeket követték az elektro-mechanikus számítógépek. Ezeket lyukkártya vezérelte és logikai kapukként reléket használtak. Ezeket a gépeket hamar felváltották az elektronikus számítógépek, amelyek már ugyanazokon az elveken működtek, mint a mai gépek.[4] Ezeket nevezzük Neumann-elveknek, amelyeket Neumann János dolgozott ki 1946-ban, és röviden összefoglalva a következőképpen fogalmazhatók meg:

- A gépnek öt alapvető funkcionális egységből kell állnia: bemeneti egység, memória, aritmetikai egység, vezérlőegység, kimeneti egység
- A gép működését a tárolt program elvére kell alapozni. Ez azt jelenti, hogy a gép a program utasításait az adatokkal együtt a központi memóriában, bináris ábrázolásban tárolja, s a Boole-algebra műveleteit ezek sorrendjében hajtja végre.[5]

A számítógépek fejlődését a minél nagyobb számolási sebesség és minél nagyobb számolási pontosság szabta meg. A soktonnás, több ezer elektroncsöves monstrumoktól, a mai

asztali illetve hordozható vagy akár szuperszámítógépekig, mind a fentebb említett sebesség és pontosság volt, amit a mérnökök szem előtt tartottak. Az elektroncsöveket felváltották a tranzisztorok, azokat pedig az integrált áramkörök. Ily módon az egyre kisebb méret egyre több alkatrész beépítését tette lehetővé és egyre nagyobb számítási kapacitást eredményezett. Azonban ez a fejlődés csak kvantitatív jellegű volt, mivel a számítógépek működési alapelvei változatlanul a Neumann-elvek maradtak.

Útkeresések a számítástechnika kihívásainak megoldására

Feladatok nehézsége

Egy feladat komplexitását az határozza meg, hogy milyen összefüggés áll fenn a bemeneti adatok mennyisége és a feladat elvégéséhez szükséges lépésszám között, ez utóbbi tulajdonképpen egyenesen arányos a megoldáshoz szükséges idővel. Az alábbi felsorolásba röviden tekintsük át a feladtok komplexitási kategóriáit:

- **P** komplexitásúnak nevezzük azokat a feladatokat, amelyek megoldásához szükséges idő egy polinomiális függvény szerint nő a bemeneti adatok számának függvényében. Ilyen feladatok például a szorzás, osztás, egy mátrix determinánsának kiszámítása.[6]
- **NP** komplexitás azt jelenti, hogy a feladat megoldása nem, de a megtalált megoldás helyességének leellenőrzőséhez szükséges idő polinomiálisan függ a megoldás méretétől. Ide tartozik minden P feladat, ám azok a feladatok is ide tartoznak, amelyek megoldása jóval több időbe kerül,mint egy P feladat, ami akár exponenciálisan is függhet a bemeneti adatmennyiségtől, de az ellenőrzés P nehézségű. Például a faktorizáció, gráf színezés, Hamilton ciklus keresése egy irányított gráfon.[7]
- **NP teljes** komplexitású feladatok a legnehezebb NP feladatok. Ha sikerülne ezekre a feladatokra polinomiális megoldást találni, akkor az azt jelentené, hogy a P és az NP nehézségű problémák halmaza ekvivalens. Ilyen NP teljes feladatok a Hamilton ciklus keresése, gráf színezés, SAT probléma (Boolean satisfiability problem, ld. lentebb).
- **NP nehéz** feladatok legalább olyan nehezek,mint bármely NP teljes feladat. Más szóval: Ha megoldunk egy NP nehéz feladatot, akkor létezik olyan NP teljes feladat, melyre a megoldás szintén igaz, úgymond sajátos esete az NP nehéznek. Példák NP nehéz feladatokra : spínüvegek energiaminimalizációja, utazó ügynök problémája, protein láncok tekeredése[8]

A jelenlegi számítógépek a P komplexitású feladatok megoldására alkalmasak optimálisan. Az ennél bonyolultabb feladatok megoldásához új megoldásokat kell illetve kellett kerseni.

Az előző fejezetben röviden áttekinthettük a számítástechnika fejlődésének klasszikus

útját. Eddig a történet meglehetősen lineárisnak bizonyult, egymást követték a különböző elvi és technikai megoldások. Jelenleg több út is kínálkozik a folytatásra, ezekből láthatunk néhányat ebben a fejezetben.

Erősen párhuzamos számítások klasszikus számítógépeken

Bonyolult feladatok egyik megoldási lehetősége a feladat részfeladatokra történő felbontása, és ezen részfeladatok kiosztása egy-egy processzornak. Ilyen módon működnek a számítógép klaszterek és a több magos processzorok illetve a több processzoros alaplapok. A mai szuperszámítógépek is ilyen felépítésűek.

Ilyen számítógépeket eredményesen alkalmaznak atomi rendszerek, makromolekulák és biomolekulák dinamikájának a realisztikus szimulációjára; törések, töredezések, plaszticitás modellezésére; granuláris anyagok viselkedésének szimulációjára, és még sok más területen is.

Azonban hátránya is van az ilyen elrendezésnek az előnyei mellett. A hűtési igények miatt viszonylag kevés processzor helyezhető egyazon alaplapra. Így viszont sok idő vesz el a különböző alaplapokon levő processzorok közti kommunikációra. Tehát az eredő számítási kapacitás nem lesz egyenlő a processzorok számítási kapacitásának összegével a kommunikációs időveszteségek miatt.

Grid és Internet számítások

A számítógépklaszterekhez hasonlóan, a Grid és Internet számítások is részfeladatokra, párhuzamos és megosztott számításokra bontják a komplex feladatokat, és szétosztják azokat több számítógép között. A számítógépek azonban lehetnek bárhol, csupán az internethez kell legyenek kapcsolva. Ezen számítási módszereknek a célja az interneten és a kisebb lokális hálózatokban levő számítógépeknek a kiaknázatlan szabad CPU idejét kihasználni, ami nagyon nagy számítási potenciált rejt magában. Ilyen módon akár több százezer processzoros virtuális szuperszámítógép is nyerhető. A hálózaton keresztül, legyen az lokális vagy az internet, a Grid-ben résztvevő számítógépek megkapják feladataikat és azon dolgoznak, majd visszaküldik az eredményeiket.

Nagy adathalmazok feldolgozására alkalmazzák, ám a számítógépek összekötöttsége közelről sem optimális, ezért nem mindig a leggyorsabb megoldást eredményező stratégia a legoptimálisabb a számítógépek közti feladatok kiosztására.

Ilyen módon próbálja feldolgozni a SETI az űrből felfogott hatalmas jelmennyiséget, önkéntesek bevonásával, akik egy kliens programot futtatnak a számítógépükön, így annak szabad processzoridejét a megkapott adatcsomagok feldolgozására használva ki.[9]

Parazita számítások

Az interneten használt TCP kommunikációs protokollt használja ki, hogy sok számítógépen oldja meg a SAT (Boolean satisfiability problem) feladatot, ami egy NP teljes feladat. A feladta lényege, hogy ha adott egy bináris formula, azt kellene eldönteni, hogy léteznek-e olyan változók, amelyekre ez a formula igaz lesz.
Kihasználva a CHEKSUM utasítást, az internetre kötött számítógép, amelynek elküldik a feladatot, leellenőrzi, hogy teljesül-e a formula. Ha igen, akkor a számítógép visszaküldi a sort a küldő számítógépnek. Ilyen módon csak a formulát teljesítő sorokat kapjuk vissza.[10]

A kvantumszámítógépek

A kvantumszámítógépek jelentik a számítástechnika nagy álmát. A kvantummechanikai jelenségek közvetlen alkalmazása révén valósulna meg az adattárolás, műveletvégzés és adattovábbítás.

Az adatokat elemi kvantummechanikai állapotokban kódolják. A számítógép maga szuperponált és kever kvantummechanikai állapotokkal dolgozik. Egy ilyen kevert állapotot nevezünk qubit-nek. Egy qubit-en rengeteg lehetséges állapotot lehet tárolni, és egy operációt egy lépésben mindezeken egyszerre lehetséges elvégezni. A qubit-nek minden lehetséges állapota egyszerre létezik valamekkora valószínűséggel. Egy művelet egy unitér transzformáció a qubitekkel jellemzett kevert kvantummechanikai állapotra. Ez úgy is felfogható, mintha a műveletet az összes tiszta állapotra egyszerre, egy lépésben végeznénk el. Ha tehát van N qubit-ünk és elvégzünk egy transzformációt, az egy lépésben 2^N műveletet jelent.

Az adattovábbítást az összefonódott kvantummechanikai állapotok valósítanák meg, ami ilyen módon egy végtelen sebességű, disszipációmentes adatátvitel lenne. Mivel az összefonódott állapotokban levő részecskepár egyik tagjának ha megváltoztatjuk a kvantummechanikai állapotát, ez azonnali változást eredményez a másik tagnál is. Ilyen értelemben végtelen az információtovábbítás sebessége.

Elméletben nagyon jó megoldásnak tűnik a kvantumszámítógép, de gyakorlati megvalósítása számtalan nehézségbe ütközik. A kvantummechanikai állapotok manipulációja, vagyis az adatbevitel és adatkivétel nagyon nehéz. A kvantummechanikai állapotokat az elemi kölcsönhatások befolyásolják és megváltoznak. A kvantumkapuk, a számítógép azon részei, amelyek a műveleteket végzik, egyelőre megvalósítatlanok. A kvantumszámítógép jó ötletnek tűnik, de még nagyon messze van a gyakorlatban alkalmazható kiviteltől.[11]

A DNS számítógép

A DNS (dezoxiribonukleinsav) molekulák tárolják az élő szervezetek genetikai információját. Tartalmazza a szervezet számára szükséges összes fehérje szintézisének

információját. A DNS-ben az információ a nukleotid bázisok sorrendjeként van tárolva. 4 nukleotid építi fel : adenin, timin, citozin, guanin (A,T,C,G). Ilyen módon az információ négyes számrendszerben van kódolva. Az adatsűrűsége óriási: 1 gramm DNS 10^{14} MB adatot hordozhat.

A DNS molekulák stabilitását növeli és a másolást megkönnyíti a nukleotid bázisok közti komplementaritás. (A – T, C – G). Az ilyen módon összekapcsolódott nukleotidok egy kettős spirálú DNS-t hoznak létre, ahol a két szál egymásnak tükörképe.

Enzimek segítségével a DNS molekula teljes egészében lemásolódhat, feldarabolódhat vagy csak bizonyos szakaszairól is készülhetnek másolatok. Ezek a másolatok összekapcsolhatók újabb lánccá vagy újabb enzimek hatására további szekvenciákra bonthatók.

Az információtároló képesség és az információkkal végezhető műveletek, mint a másolás, összefűzés, darabokra bontás és részleges másolás, elvben lehetővé teszik egy DNS számítógép megvalósítását. Egy ilyen számítógép előnye, hogy egyszerre, teljesen párhuzamosan, lehet elvégezni a műveleteket nagyon sok különböző információtartalmú DNS láncon.

Hátránya viszont, hogy minden problémamegoldáshoz más-más eljárást kell alkalmazni, ami más enzimek és más körülmények biztosítását jelenti. Az adatbevitel és kivétel körülményes. Ráadásul a DNS spontán módon is mutálódhat, ami hibákat eredményez. Egyelőre nem egy ideális jelölt a jövő számítógépe címre, de még számos érdekes lehetőség rejlik benne.

CNN (Cellular Neural Network) számítógép

Jelenleg a leginkább előrehaladott új számítástechnikai irányzatot képviselik a CNN számítógépek. Elvi felépítésük nagyon egyszerű. Egy elemi processzorokból (cellákból) álló rács, amelyben a szomszédos cellák egymással össze vannak kapcsolva. Minden cella teljesen párhuzamosan működik a többivel. Úgy a cellák bemenő jele, mint a cellák állapotát jellemző érték és a kimenő jel mind-mind analóg, folytonosan változó jel. Az időbeni diszkretizáció nincs jelen az ilyen számítógépekben. Egy cella mindenkori állapotát a cella előző állapota és a szomszédainak az állapota szabja meg. [12]

Egy ilyen rendszer un. analogiakai számítógépet alkot: analóg jelekkel dolgozik, és képes logikai műveletek elvégzésére is. Elvi felépítésükből fakadóan a feladatok nehézsége megfordul az ilyen számítógépek esetén. Könnyedén elvégezhetők a képfelismerés, képfeldolgozás, parciális differenciál egyenletek megoldásának feladatai, azonban az egyes egyszerű matematikai műveletek elvégzése nagyon bonyolulttá válik. Kitűnően alkalmazhatók valós idejű jelfeldolgozásra ami növeli robotikai jelentőségüket. Előnyeik

közé tartozik még az alacsony energiafogyasztás, és biológiailag inspiráltságukból származó lehetőségek. Jelenleg körülbelül olyan gyorsak, mint a hagyományos chipek, de rohamosan fejlődnek.

Számítástechnika kollektív viselkedést mutató rendszerekkel

A számítástechnika feladatainak megoldásában ígéretesnek tűnő irány az egymással párhuzamosan működő, egyszerű és azonos felépítésű elemekből álló rendszerek alkalmazása. Egy ilyen rendszer az elemeik közti kölcsönhatások függvényében különböző kollektív viselkedéseket mutathat, és ebben a kollektív viselkedésben nagy számítástechnikai potenciál várhat még felfedezésre.

Ebben a évben egymással párhuzamosan működő villogó áramkörök kollektív viselkedését vizsgáltam szimulációkon keresztül. Felmerült a kérdés, hogy alkalmazható lenne-e egy ilyen villogó áramkörökből felépített rendszer valamilyen számítási probléma megoldására. Mivel az általam tanulmányozott rendszerek tulajdonságai jól egyeznek a fentebb megnevezettekkel, a válasz jó eséllyel igen volt.

Feladatként kitűzött probléma a prímszámok generálása illetve egy számhalmaznak olyanképpen való szűrése, hogy a megmaradó elemek ne osszák egymást. A megvalósítás szimulációban történt.

Prímkeresés villogó áramkörökkel

Az első n darab prímszám generálását n darab globálisan csatolt villogó áramkör segítségével oldottam meg és annyi időlépésben, amekkora az n-edik prím értéke.

Egy villogó áramkör tulajdonságai a következők: rendelkezik egy saját t periódussal amit maga állít be egyszer és csak egyszer és utána tartja azt, amíg a rendszer működik. Minden periódus végén egy egységnyi (időlépésnyi) hosszúságú fényimpulzust bocsát ki. Annak eldöntésére, hogy hányadik időlépésnél tart, rendelkezi egy f fázisszámlálóval. Minden áramkör képes érzékelni a többi áramkör által kibocsátott fényt, mindaddig, amíg le nem rögzíti a periódusát.

A rendszer tulajdonképpen az Eratosztenész szitájaként ismert algoritmust követi. Ami röviden a következő. Felírjuk a természetes számokat 2-től kezdődően egy tetszőlegesen megválasztott értékig, növekvő sorrendben. Vesszük az első számot, ami jelen esetben a 2, és kihúzzuk a táblázatból az összes többszörösét. Ezután vesszük a növekvő sorrendben a következő nem kihúzott számot (a 3-at) és kihúzzuk ennek is a többszöröseit. Ezt ismételve a táblázatban végül megmaradó számok a prímszámok lesznek a 2 és a megválasztott felső érték között.

A villogó áramköröket arra használtam fel, hogy automatikusan generálják a

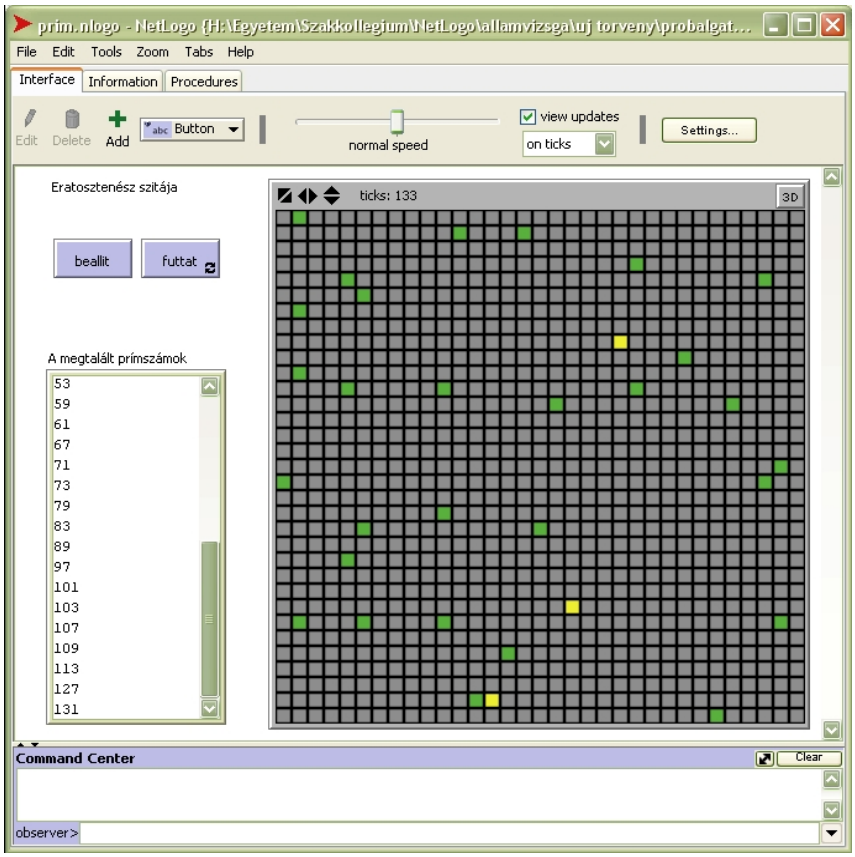
prímszámok többszöröseit. Egyetlen áramkör úgy működik, hogy amíg fény van a rendszerben nem rögzíti le a periódusát, csupán számolja, hogy hány időlépés telt el a rendszer indulása óta. Amint a rendszerben lesz egy olyan időlépés, amelyben nem világít egyetlen addig aktiválódott áramkör sem, akkor az eddig csak „magában számoló” áramkör kigyúl és rögzíti a periódusát, annak annyi időlépésnyi hosszat adva, mint amennyi eltelt a rendszer indítása óta. És ez az érték mindig a sorrendben következő prím lesz. Ezután már csak ennek a periódusnak a többszöröseit jelentő időlépésekben fog kigyúlni jelezve a prím többszöröseit.

A rendszer kiindulási állapotában minden áramkör a fázisszámlálója a 2-es értéket kapja, azaz a rendszer a második időlépéstől indul. Mivel egyetlen áramkör sem világít, ki fog gyúlni egy áramkör, ami majd ezután mindig ki fog gyúlni minden második időlépésben. Annak érdekében, hogy az összes áramkör ne gyúljon ki már a rendszer indulásakor, minden áramkör rendelkezik egy $v \in (0,1)$, véletlenszerűen megválasztott várakozási konstanssal, ami azt jelenti, hogy miután az áramkör érzékeli, hogy nincs fény a rendszerben, még egy *v* -nyi időt vár,és ha addig sem kapott bejövő fényjelet, akkor gyúl ki és rögzíti a periódusát.

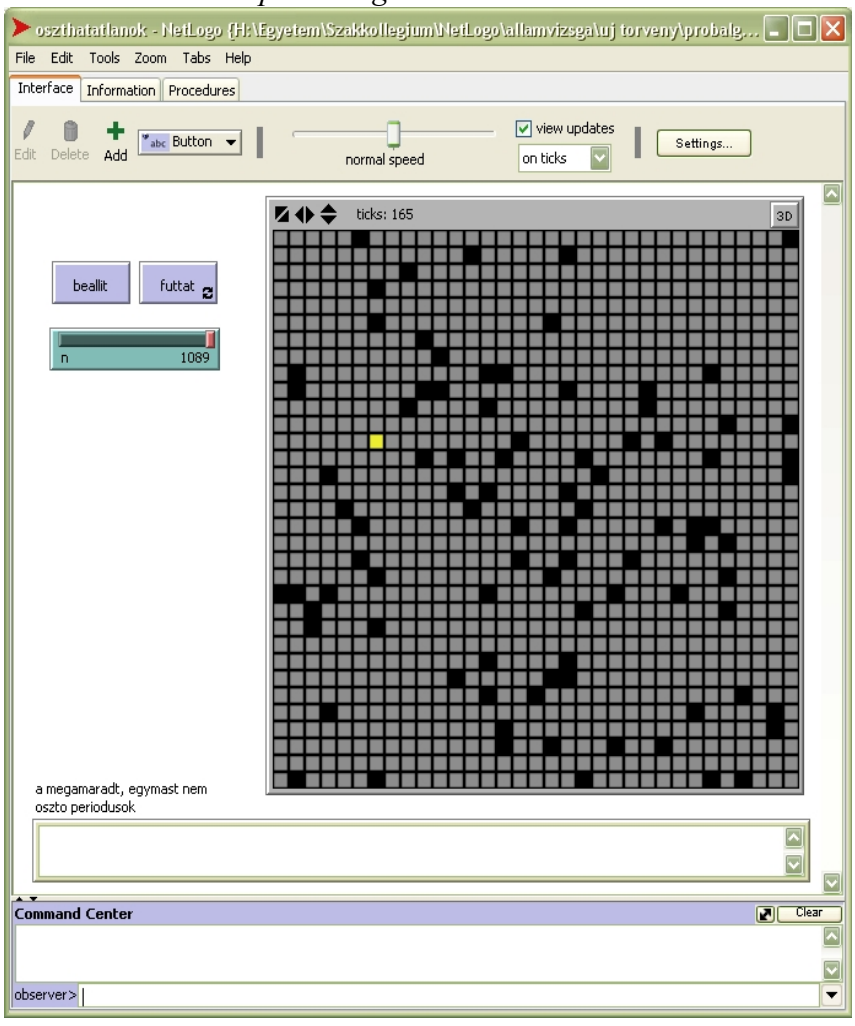
Az általam írt szimuláció a NetLogo programozási környezetben készült. A NetLogo egy modellezésre kifejlesztett, ingyenes, objektum-orientált nyelv. Nagy előnye a könnyű grafikus megjelenítés és az egyszerű applet készítési lehetőség, ami lehetővé teszi az elkészített modellek honlapokba való beépítését.

A szimuláció egy egyszerű párbeszédablakként jelenik meg (1. kép). Két funkciógomb van az ablakban. A „beallit” gomb megnyomása létrehozza az *n* darab, jelen esetben 1089 darab, áramkörmodellt, és beállítja a kezdőfeltételeket. A „futtat” gomb lenyomása elindítja a prímgenerálást a fent leírt módon. Az ablakban található fekete hátterű mezőben levő négyzetecskék jelölik a villogó áramköröket. A színük mutatja az illető áramkör állapotát. A szürke a még nem rögzített periódusú, zöld a rögzített periódusú a sárga az éppen világító áramkört jelöli. A baloldali listába pedig a megtalált prímeket íratom ki. A szimuláció addig fut, amíg minden áramkörnek lesz egy lerögzített periódusa. Ezen időlépések száma pedig egyenlő az *n*-edik prím értékével.

A szimuláció forráskódját a függelékben csatolom prim.nlogo néven.



1.kép – Prímgenerálás szimuláció



2.kép – Az egymást nem osztó elemek kikeresése

Egy halmazból az egymást nem osztó számok kikeresése

Hasonlóan a prímkereséshez, itt is globálisan csatolt villogó áramköröket alkalmaztam. Az áramkörök annyiban különböznek az előzőektől, hogy periódusuk előre meghatározott. Minden halmazbeli számnak megfelel egy áramkör periódusa. Ez aztán nem változik a rendszer működése során. Minden áramkör a 0 fázisból indul, egyszerre. Az áramkörök csak egyszer érzékelik a többi áramkörtől érkező fényimpulzusokat, éspedig akkor, amikor először érnek a periódusuk végére. Ha abban az időlépésben más áramkör is világít, az azt jelenti, hogy annak periódusa osztja az első felvillanását végrehajtó áramkör periódusát. Ebben az esetben az áramkör, amelynek a periódusa oszhtató volt, megszünteti további működését. Ha viszont nem volt vele egyszerre világító áramkör az első periódusa végén, akkor a továbbiakban folyatja a működését és érzéketlen lesz a bejövő impulzusokra. Ilyen módon a rendszerben csak azok az áramkörök maradnak meg, amelyek periódusai nem osztják egymást.

A szimuláció szintén NetLogo-ban készült (2.kép). A „beallit” gomb lenyomásával létrehozzuk az „n” csúszkával beállított számú áramkört, és osztjuk ki az áramkörök periódusait. Az egyszerűség kedvéért a szimulációban véletlen számokat használtam, de könnyen lehet célzottan kiválasztott számokat is periódusértéknek tenni. A „futtat” gomb lenyomásával elindul a szimuláció. A fent leírt módon kiszűrődnek azok a periódusértékek, amelyek nem osztják egymást. A szürke négyzetek jelölik az alapállapotban lévő áramköröket, a sárgák az éppen világító áramköröket, a fekete mezőkön nem található áramkör.

Következtetések

Láthattuk, hogy a számítástechnikában a fejlődés és a komplex feladatok megoldásának érdekében paradigmaváltásra van szükség. Egy lehetséges út a kollektív viselkedést mutató rendszerek alkalmazása valamilyen számítási feladat megoldására. Az általam kifejlesztett prímgeneráló algoritmus jó példa a párhuzamosan működő egyedekből álló, kollektív viselkedést mutató rendszerek számítástechnikai alkalmazására. Kifejezetten hatékony módon generálja le az első n darab prímszámot , mivel az n-edik prímszám megtalálásához szükséges időlépések száma csupán annyi, mint amekkora az n-edik prím értéke. Hasonlóan, az egymást nem osztó számok kikeresésének algoritmusa is annyi időlépésben végzi el ezt a szűrést a halmazon, amekkora a legnagyobb elem értéke a halmazban, a halmazelemek számától függetlenül. Hátrányuk viszont, hogy minden értéket egy áramkör kell képviseljen, ami nagy anyagigényt jelent. Viszont ezek az elemek nagyon

egyszerűek, mondhatni olyan egyszerűek, mint az első összeadógépek fogaskerekei.

A bemutatott két példa igazolja, hogy hatékony algoritmusokat lehet találni ilyen típusú rendszerek alkalmazásával bizonyos feladatok megoldására, vagyis ez egy olyan út, amit érdemes a jövőben is vizsgálni.

Irodalomjegyzék és hivatkozások

1. http://hu.wikipedia.org/wiki/Moore_törvény
2. http://www.ttk.pte.hu/ami/phare/tortenet/Pascal.html
3. http://www.ttk.pte.hu/ami/phare/tortenet/Leibniz.html
4. http://hu.wikipedia.org/wiki/A_számítógép_története
5. http://hu.wikipedia.org/wiki/Neumann-elv
6. http://en.wikipedia.org/wiki/P_(complexity)
7. http://en.wikipedia.org/wiki/NP_(complexity)
8. http://en.wikipedia.org/wiki/NP-hard
9. http://hu.wikipedia.org/wiki/Grid_computing
10. Parasitic computing - Albert-László Barabási, Vincent W. Freeh, Hawoong Jeong & Jay B. BrockmanNature **412**, 894-897 (30 August 2001)
11. http://hu.wikipedia.org/wiki/Kvantumszámítógép
12. L. O. Chua , L. Yang, IEEE Transactions on Circuits and Systems 35. No. 10, 1988

Függelék

prim.nlogo

```
turtles-own [t f v]

to beallit
clear-all
set-default-shape turtles "square"
ask patches [sprout 1 [set color gray
                       set v random-float 1]]
tick
tick
end

to futtat
ask turtles with [color != gray] [set f (f + 1) mod t
                                  ifelse (f = 0) [set color yellow ]
                                                 [set color green] ]
if not any? turtles with [color = yellow]
          [ask turtles with [v = min [v] of turtles with [color = gray]][set t ticks
                                                                         set color yellow
                                                                         output-print t]]
tick
if not any? turtles with [color = gray] [stop]
end
```

oszthatlanok.nlogo

```
turtles-own [t f n_villanas]

to beallit
clear-all
set-default-shape turtles "square"
ask n-of n patches [sprout 1 [set color gray
                              set t (random 1500) + 2
                              set n_villanas 0]]
end

to futtat
ask turtles [set f (f + 1) mod t
             ifelse (f = 0) [set color yellow
                             set n_villanas n_villanas + 1]
                             [set color gray]]

if count turtles with [color = yellow] > 1
        [ask turtles with [n_villanas = 1 and color = yellow][die]]
tick
ifelse any? turtles [if ticks > max [t] of turtles [output-print sort [t] of turtles
                                                    stop]]
             [stop]
end
```

Tartalomjegyzék