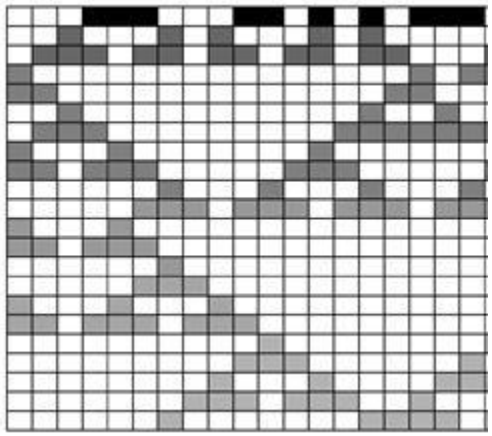

CELLULÁRIS AUTOMATA (SEJTAUTOMATA)

Számítások randevú típusú szinkronizálással

RÖVID ÖSSZEFOGLALÁS

A sejtautomata leggyakoribb formája: egy négyzetrácsban (a *sejttér*ben) helyezkedik el, a négyzetrácsok által közrefogott cellákat sejteknek nevezzük. A sejteknek különféle állapotaik lehetnek (véges sokféle). Ahogy az idő telik, a cellák változtatják állapotukat, általában saját és más sejtek, például néhány szomszédjuk előző időpillanatbeli állapotától függően.

Burian Sándor , AWXYHE

Szoftverfejlesztés párhuzamos architektúrákra

2020-2021, első félév

Óbudai Egyetem, Neumann János Informatikai kar

Tartalom

Bevezetés, a problémabemutatása

“A feladatteret cellákra osztjuk. Minden cella véges állapotok közül pontosan egyet vesz fel. A cellákra a szomszédjai valamilyen hatással vannak bizonyos szabály szerint, és minden cella egy „generációs” szabállyal is rendelkezik, amely szimultán fut. A szabályok újra és újra alkalmazásra kerülnek minden generációs lépésben, így a cellák fejlődnek, vagy megváltoztatják állapotukat generációról generációra. A legismertebb celluláris automata John Horton Conway cambridge-i matematikus „Élet játéka” (“Game of Life”).” [1]

Az élet játéka

A táblás játék elméletben végtelen méretű kétdimenziós tömbökkel. Minden tömb cellákat tartalmaz. Tehát minden cellának 8 szomszédja van.

Pl:

	⋱	…	…	…	⋰
tömb	…	Cella	Cella	Cella	…
tömb	…	Cella	Cella	Cella	…
tömb	…	Cella	Cella	cella	…
	⋰	…	…	…	⋱

A cellákban amikben egy-egy organizmus lehet. Kezdetben néhány cella fogallat, a következő szabályok szerint, amiket Conway fektett le [1]:

1. Minden organizmus aminek 2-3 szomszédja van életben marad a következő generációra.
2. Minden olyan organizmus amelynek 4+ szomszédja van meghal a túlnépesedés miatt.

[1] https://elearning.uni-obuda.hu/main/pluginfile.php/274206/mod_resource/content/1/P%C3%A1rhuzamos_programoz%C3%A1s_javasolt_feleves_feladatok.pdf hozzáférés dátuma: 2020 szeptember 28

3. Mindegyik olyan organizmus amelyiknek 2-nél kevesebb szomszédja van, tehát maximum egy, meghal az izoláltság miatt.
4. Minden üres cellában amelynek pontosan három nem üres szomszédja van egy új organizmus születik.

A példa mögé több mindent is beleláthatunk, a cellák tartalmai könnyen tovább gondolhatóak nyulakra és rókákra egy szigeten, vagy három dimenziós tömbökkel halakra és cápákra, az alábbiak szerint [1]:

Halak:

1. Ha van üres szomszédos cella odaúszik, ha több cella is üres akkor véletlenül választ egyet
2. Ha nincs üres szomszédos cella akkor helyben marad
3. Ha mozog és eléri egy természetes értékként megadott nemzési idejét akkor egy újabb hal kerül a szomszédos, üresen maradt cellába
4. Adott idő után a hal elpusztul

Cápák:

1. Ha szomszédos cellában hal van, akkor oda lép és megeszi, ha több szmszédos cellában is hal van akkor véletlen szerűen választ egyet.
2. Ha nincs szomszédos cellában hal akkor egy szomszédos üres cellába mozog, mint egy hal, ha több szomszédos cella is üres akkor véletlenül választva.
3. Ha eléri a nemzési idejét akkor akár egy hal új cápa kerül egy megüresedő cellába.
4. Ha a cápa adott generáción keresztül nem eszik akkor elpusztul.

Hasonlóan mögé lehet látni folyadék vagy gáz dinamikai folyamatokat, biológiai fejlődést, légáramlás vagy eróziós folyamatok modellezését.

Párhuzamosítás kérdése

Mivel egy nagy területen szükséges sokszor ugyanazt a műveletet elvégeznünk (*történetesen, hogy a különböző szomszédjaiban mik az állapotok egy-egy cellára nézve*) ezért a párhuzamosítás használata egyértelműen hasznos. Ugyanakkor az is egyértelmű, hogy időnként szinkronizációra van szükség. Elképzelésem szerint a teret

felosztom nagyjából egyenlő részekre, annyira ahány magra optimalizálva számítom a feladatot[2]. Innentől csupán az egymással szomszédos területeket számoló folyamatok kell megvárják egymást, és csak azok kell szinkronizálódjanak, így, globális szinkronizációra nincs is igazán szükség, hiszen a terület két teljesen eltérő sarkában számolt értéknek nem sok értelme van, hogy egymásra várjon, mert nem sok közük van egymáshoz, nem befolyásolja egyik a másikat, a szinkronizálás pedig alapvetően egy költséges művelet [2], így a minnél kevesebb szinkronizálás a praktikusabb a gyorsaság szempontjából.

"Az egyes folyamatok csak arra várnak, hogy az alapgráfbeli összes szomszédjuk *helyi_szint*-je legyen legalább *k*. Így ezt az összes folyamat maga felderítheti a *helyi_szint* minden egyes növekedése után a szomszédba vezető éleken küldött üzenettel. […] időbonyolultságának felső határa:

$$O(\mathrm{n}\log \mathrm{n}(\mathrm{l}+\mathrm{d}))$$

A kommunikációs bonyolultság azonban rosszabb a mindenegyes szinten használt szinkronizációs üzenetek miatt. Ez most $O(|\mathrm{E}|\log \mathrm{n})$ "[3]

Ezek persze csak a várt eredmények. A szükséges tárhely előre nem leszögezhető, hiszen a feladat elviekben végtelen tereket használ, így végtelen tárhelyre is volna szükség, ami persze jelenleg nem lehetséges. Előre tervezetten vagy a felhasználó által beadható, vagy a teljes memóriát hkihasználó tárhelyre lehetne felkészülni mint kényelmes-praktikus számítási modell.

[2] Ez akár dinamikusan is lekérhető, így a program konkrét processzortípustól függetlenül is optimalizált lehet [1]

Forrásjegyzet

Borítókép: [Online] Available: https://regi.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0063_13_parhuzamos_algoritmusmodellek/images/fig3-9.jpg. [Hozzáférés dátuma: 2020 Szeptember 28]

[1] Environment.ProcessorCount Property [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.environment.processorcount?redirectedfrom=MSDN&view=netcore-3.1#System_Environment_ProcessorCount [Hozzáférés dátuma: 29 09 2020]

[2] TÖBBSZÁLÚ/TÖBBMAGOS PROCESSZORARCHITEKTÚRÁK PROGRAMOZÁSA [Online] Available: https://elearning.uni-obuda.hu/main/pluginfile.php/274151/mod_resource/content/1/0053_Tobbszalu_Tobbmagos_Processzorarchitekturak_Programozasa.pdf [Hozzáférés dátuma: 29 09 2020]

[3] Nancy Ann Lynch: Oszott algoritmusok, ISBN: 963-9301-03-5 Kiskapu kiadó, 2020 árilis 18, 479. oldal