Celularni avtomati (Game of Life)

Matej Kristan in Tadej Hiti

Motivacija:

Izvor motivacije za izbiro igre “Game of Life” ter celularnih avtomatov splošneje kot projekt pri predmetu Paralelni sistemi izhaja iz zanimivih ter abstraktnih konceptov, ki se jih teoretično ozadje igre dotika. Zanimivo je, da kljub znanim pravilom obstoja celic skozi generacije ne obstaja algoritem, ki bi mu za vhod podali vhodno (začetna generacija) in poljubno (poljubna generacija) stanje, ta pa bi izračunal ali bo do tega poljubnega stanja oziroma generacije sploh prišlo. Prav tako naju je navdihnilo spoznanje, da v naravi obstajajo biološki procesi oziroma vzorci, ki se jih da dobro simulirati s pomočjo celularnih avtomatov.

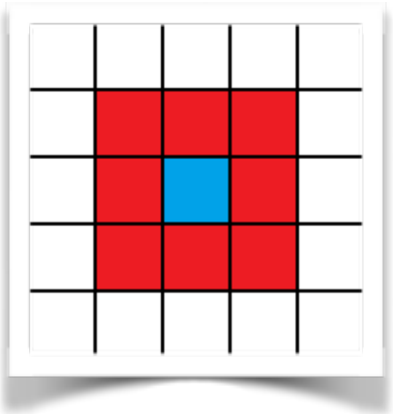
Opis:

Povod nastanka igre Game of Life prihaja iz prejšnjega stoletja, ko je britanski matematik John Horton Conway želel rešiti zastavljeni problem matematika John von Neumann-a, ki je želel odkriti ali obstaja hipotetični stroj, ki bi bil zmožen na podlagi zbranih surovin zgraditi kopije sebe ter tako pospešiti iskanje možnosti obstoja življenja ter kolonizacije onkraj našega planeta.

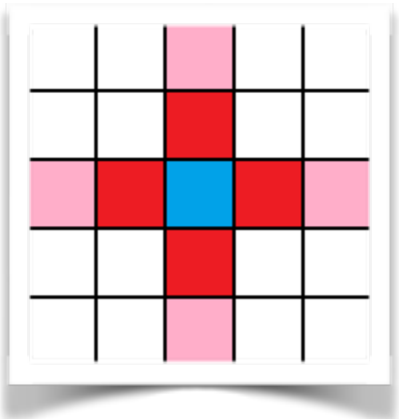
Game of Life je celularni avtomat, je igra brez igralca, kar pomeni da je evolucija odvisna od nič drugega kot začetnega podanega stanja ter vnaprej specificiranih nespreminjajočih pravil, ki opisujejo spreminjanje kvadaratnih celic na dvo-dimenzinalni neskončni ortogonalni mreži.
Vsaka celica na mreži drži eno izmed dveh stanj: živo ali mrtvo. Vsaka celica je s pomočjo vnaprej definiranih pravil v interakciji z sosednjimi celicami, ki držijo usodo trenutne celice ali bo ta skozi naslednjo generacijo obstala ali ne.

Najina realizacija problema bo priredba, kar pomeni da se bova osredotočila tudi na druge celularne avtomate in ne samo celularnega avtomata znanega kot “igra življenja”, kar pa pomeni da sva definirala več možnih soseščin, ki vplivajo na obstoj celice skozi generacijo.

Med slednjimi sva trenutno implementirala sledeči soseščini celic, kjer je z modro ponazorjena opazovana celica in z rdečo sosednje celice, ki vplivajo na opazovano.



Moorova soseščina



von Neumannova soseščina

Pravila:

- Vsaka živa celica z manj kot dvema živima sosedoma umre kot posledica premajhne populacije.
- Vsaka živa celica z dvema ali tremi živimi sosedi preživi skozi naslednjo generacijo.
- Vsaka živa celica z več kot tremi živimi sosedi umre kot posledica prevelike populacije.
- Vsaka mrtva celica z natančno tremi živimi sosedi se spremeni v živo celico, kot posledica reprodukcije

Pseudokoda:

```
1   foreach (cell in grid) {
2       neighbours = 0;
3       foreach (neighbour in neighborhood) {
4           if (isAlive(neighbour))
5               neighbours++;
6       }
7       if (cell is alive) {
8           if !((neighbours == 2) || (neighbours == 3) )
9               kill cell;
10      }
11      else {
12          if (neighbours == 3)
13              revive square;
14      }
15  }
```

Ocenitev zahtevnosti algoritma:

Časovna zahtevnost: f(n,m,s) = **O(n*m*s)**
Prostorska zahtevnost: g(n,m) = O(2*(n*m)) = **O(n*m)**

n = velikost horizontalnega polja
m = velikost vertikalnega polja
s = število vplivnih okrožnih sosedov

Reference:

- https://en.wikipedia.org/wiki/Cellular_automaton
- https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
- https://www.youtube.com/watch?v=R9Plq-D1gEk

Poročilo - Sekvenčni algoritem (Game of Life)

Kratek opis:

Pri implementaciji sekvenčnega algoritma sta bila v uporabi zgolj navaden urejevalnik teksta ter terminal s pomočjo katerega sva poskrbela za zagon algoritma preko gcc - GNU prevajalnika.
Za izris generacij 'igre življenja' je bila uporabljena zunanja knjižnica SDL2 - https://www.libsdl.org/.

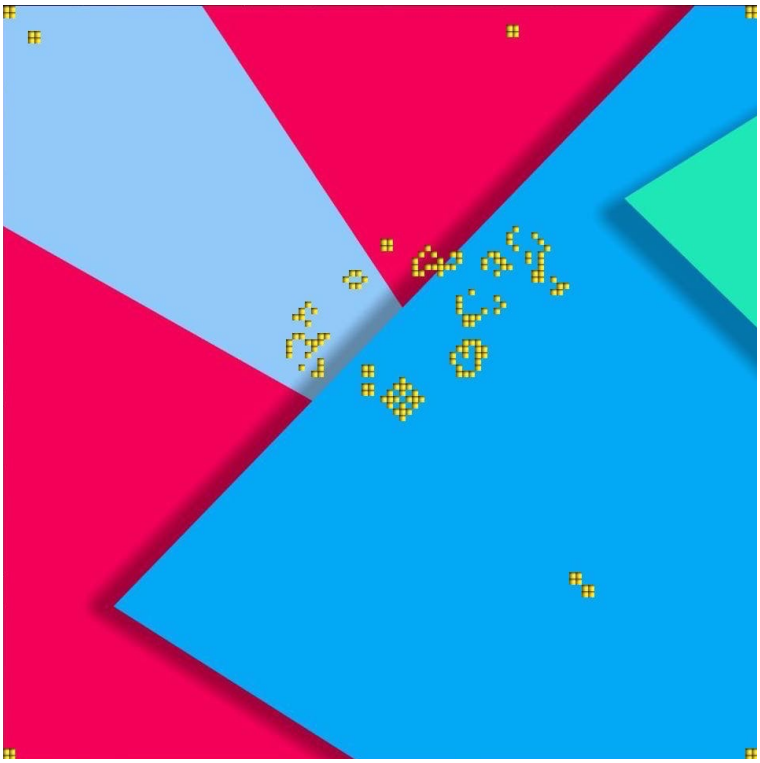
V uporabi ni bilo posebnih podatkovnih struktur, zgolj uporaba dvo-dimenzinalnih tabel med drugim tudi za predstavitev sveta algoritma.

Vhodni podatki programa:

- velikost sveta (višina, širina, in število simuliranih generacij)
- začetna generacija sveta

Meritve algoritma:

Meritve algoritma so bile izvedene na algoritmu celularnega avtomata Game of Life, pravila ostajajo nespremenjena, definirana zgoraj v podpoglavju "Pravila", torej meritve so bile izvedene na 3x3 soseščini, vsaka celica ima 9 sosedov, oziroma po principu Moorove soseščine.



Slika 1: Simulacija algoritma Game of Life

Meritve:

Tabela 1: Čas izvajanja in standardna napaka meritve (SE) v odvisnosti od velikosti reševanega problema (N = višina * širina * število simuliranih generacij).

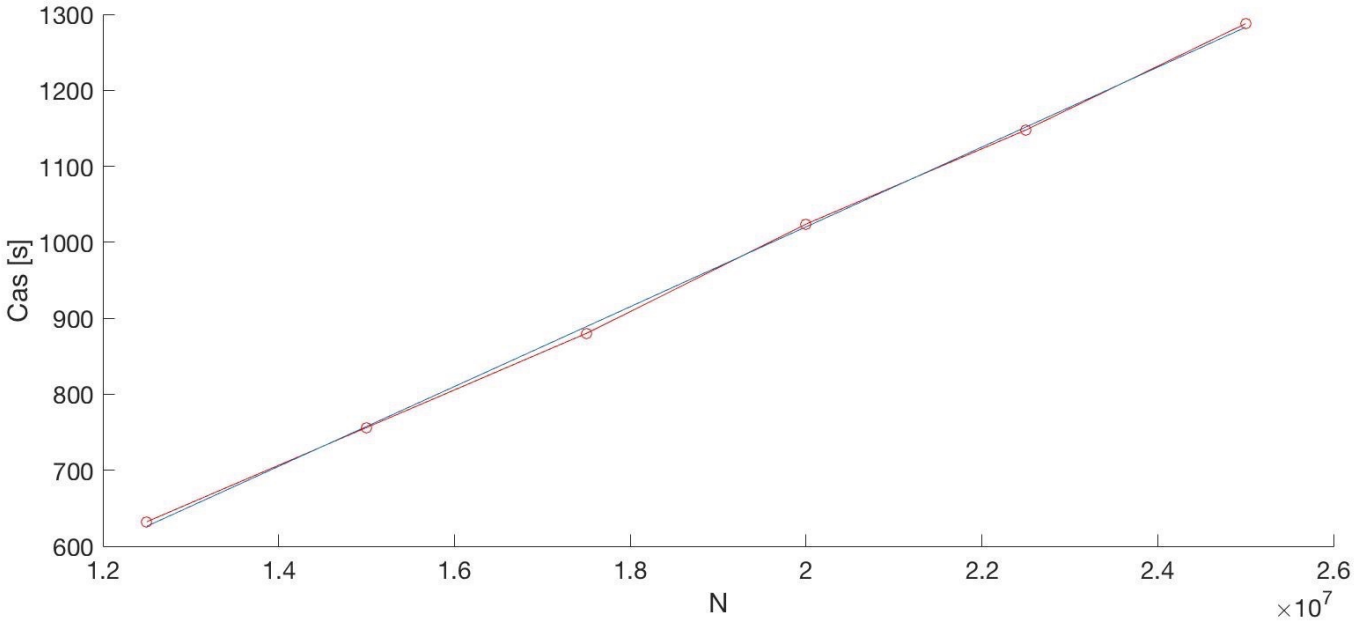
N (višina, širina, št. generacij)	Čas [s]	SE [s]
12500000 (500, 500, 50)	0.64	0,007
15000000 (600, 500, 50)	0.74	0,007
17500000 (700, 500, 50)	0.86	0,004
20000000(800, 500, 50)	0.98	0,006
22500000(900, 500, 50)	1.11	0,008
25000000(1000, 500, 50)	1.23	0,004

Tabela 2: Čas izvajanja in standardna napaka meritve (SE) v odvisnosti od velikosti reševanega problema (N = višina * širina * število simuliranih generacij).

N (višina, širina, št. generacij)	Čas [s]	SE [s]
12500000 (500, 500, 50)	0.63	0,001
15000000 (500, 600, 50)	0.76	0,004
17500000 (500, 700, 50)	0.88	0,004
20000000(500, 800, 50)	1.02	0,004
22500000(500, 900, 50)	1.15	0,003
25000000(500, 1000, 50)	1.29	0,010

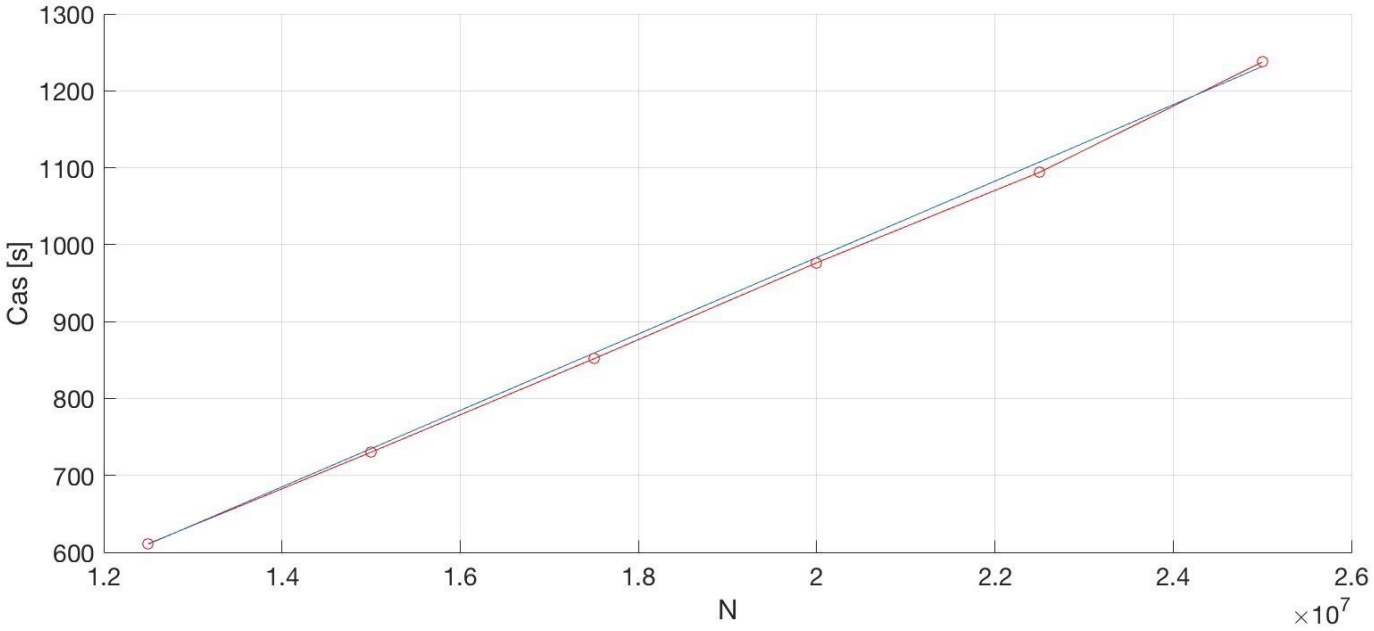
Tabela 3: Čas izvajanja in standardna napaka meritve (SE) v odvisnosti od velikosti reševanega problema (N = višina * širina * število simuliranih generacij).

N (višina, širina, št. generacij)	Čas [s]	SE [s]
12500000 (500, 500, 50)	0.61	0,002
15000000 (500, 500, 60)	0.73	0,001
17500000 (500, 500, 70)	0.85	0,001
20000000(500, 500, 80)	0.98	0,003
22500000(500, 500, 90)	1.11	0,002
25000000(500, 500, 100)	1.24	0,013



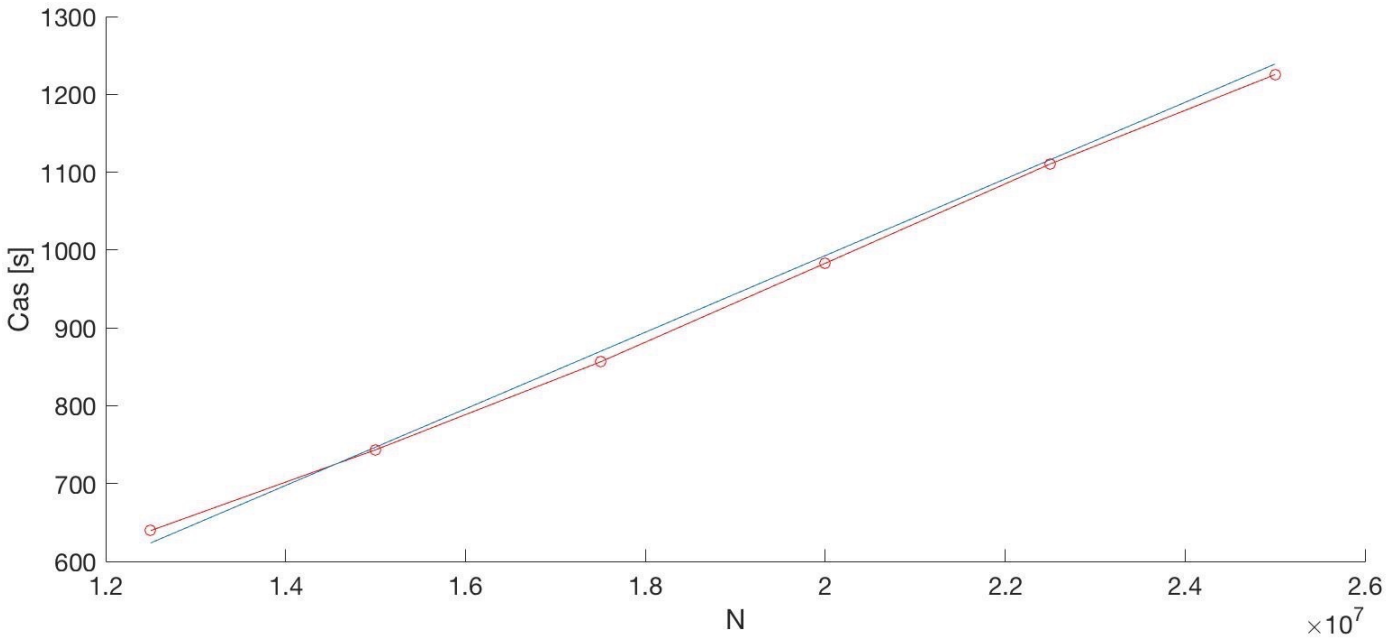
Graf 1: Čas izvajanja v odvisnosti od velikosti reševanega problema (povečevanje višine simuliranega sveta).

Izmerjena časovna odvisnost: O(y) = 8.42 + 4.92 * x



Graf 2: Čas izvajanja v odvisnosti od velikosti reševanega problema (povečevanje širine simuliranega sveta).

Izmerjena časovna odvisnost: O(y) = - 3.01 + 5.25 * x



Graf 3: Čas izvajanja v odvisnosti od velikosti reševanega problema (povečevanje števila generacij simuliranega sveta).

Izmerjena časovna odvisnost: $O(y) = -1.47 + 4.96 * x$

Teoretična časovna odvisnost:

Teoretična časovna odvisnost je za vse grafe enaka, ta je O(n) in se pričakovano zelo lepo prilega izmerjeni časovni odvisnosti.

Poročilo - PThreads algoritem (Game of Life)

Kratek opis paralelizacije algoritma:

Algoritem Game of Life z rahlo spremembo pravil ter na soseščini 5x5 je bil paraleliziran s pomočjo knjižnice PThreads, kjer vsaka kreirana nit dobi določen del polja, ki ga nit obdela ter vrne svoj rezultat, ki je kasneje združen z neko določeno 'glavno' nitjo v skupni rezultat, tako se s pomočjo delitve dela med nitmi algoritem znatno pohitri.

Delitev dela med nitmi:

Torej imamo polje velikosti W x H, kjer je *W* širina ter *H* višina obdelovalnega polja ter *p* zaznamuje število kreiranih niti.

Vsaka izmed kreiranih niti dobi določeno število vrstic polja po sledečem pravilu:

delta = H / p + 1;

nit.začetek = delta (izjema je začetna nit, kjer je začetek seveda enak 0)
nit.konec = nit.začetek + delta

Vsaka nit tako dobi enako velikost obdelovalnega podproblema, seveda so tu izjeme:

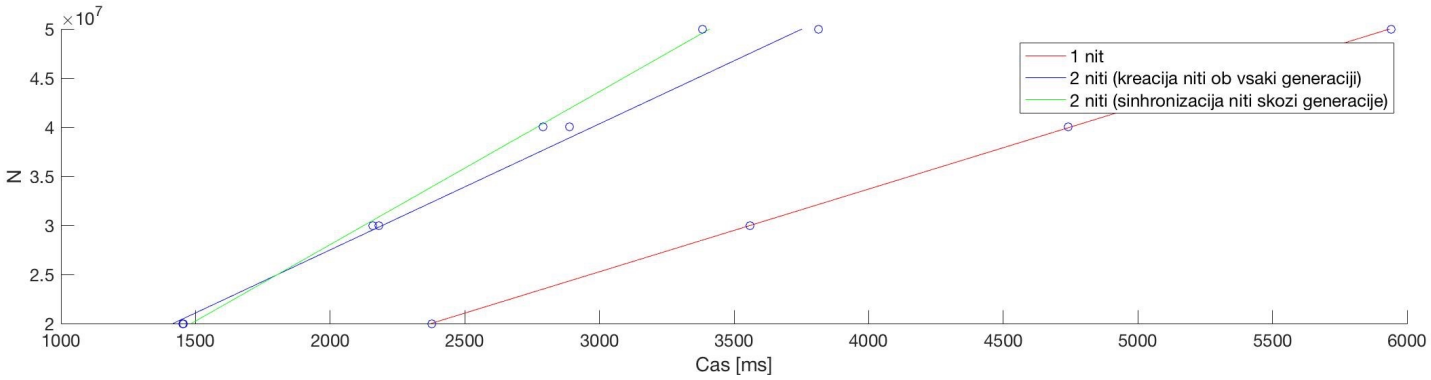
- V kolikor je število niti manj kot pa je višina obdelovalnega problema, zadnjih *H - p* ne dobi obdelovalnega kosa polja.
- V kolikor je H liho število, potemtakem dobi zadnja nit prostorsko manjši obdelovalni podproblem.

Komunikacija med nitmi:

Torej delitev dela med nitmi se izvede znotraj vsake računane generacije. Preden niti sploh dobijo kakršno koli delitev dela, se morajo počakati, da so vse zaključile z izvajanjem svojega prejšnjega podproblema (računanje prejšnje generacije). Slednje zagotovimo z ukazom *pthread_barrier_wait (pthread_barrier_t barrier)*.
Ko so niti sinhronizirane se prične dodeljevanje dela ter obdelava posameznih podproblemov. Zopet za zagotovitev, da so vse niti prenehale z obdelovanjem svojega podproblema, uporabimo *"bariero"*, ki počaka na dokončanje izvajanje vseh niti. Ko je sinhronizacija zopet usklajena začetna nič (p == 0) prične s kopiranjem vsake vrstice polja v istoležne vrstice novega polja, ki se uporabi v naslednji iteraciji.

Meritve paraleliziranega algoritma:

Vse meritve algoritma so bile izvedene na soseščini velikosti 5x5 na polju velikosti 100x100 z naraščajočim številom generacij. V zgornjih dveh podpoglavjih (Delitev dela med nitmi, Komunikacija med nitmi) sem opisal način dela ter komunikacijo med nitmi, ki ga trenutno uporabljava, sva pa tudi izvedla meritve algoritma, kjer sva skozi vsako generacijo kreirala niti ter jim dodelila podproblem na enak način, ter jih ob koncu izvajanja pokončava.
Vsaka točka na grafu predstavlja povprečno izmerjeno meritev desetih posameznih merjenj pri dani velikosti reševanega problema.



Graf 1: Čas izvajanja algoritmov v različnih načinih ter na različnem številu niti v odvisnosti od velikosti reševanega problema.

N / Čas [ms]	G1	G2	G3
2000 * (100*100)	2378	1453	1455
3000 * (100*100)	3558	2180	2157
4000 * (100*100)	4741	2888	2790
5000 * (100*100)	5940	3813	3384

Tabela 1: Časi izvajanja v odvisnosti od velikosti reševanih problemov pri različnih načinih izvajanja (glej opredelitve pod grafi)

N - velikost reševanega problema = število generacij * (W * H)
G(i) - Graf [1-3]

Specifikacije testnega sistema - podatki CPE:

Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit
Byte Order:	Little Endian
CPU(s):	4
On-line CPU(s) list:	0-3
Thread(s) per core:	2
Core(s) per socket:	2
Socket(s):	1
NUMA node(s):	1
Vendor ID:	GenuineIntel
CPU family:	6
Model:	69
Model name:	Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz
Stepping:	1
CPU MHz:	1652.726
CPU max MHz:	3100,0000
CPU min MHz:	800,0000
BogoMIPS:	5188.34
Virtualization:	VT-x
L1d cache:	32K
L1i cache:	32K
L2 cache:	256K
L3 cache:	4096K
NUMA node0 CPU(s):	0-3

Analiza meritev

S = T(sek) / T(p)
E = S / p

S - pohitritev algoritma, kjer je T(sek) čas izvajanja sekvenčnega programa in T(p) čas izvajanja paraleliziranega programa nad p nitmi.

E - Učinkovitost algoritma

	2 niti - A		2 niti - B	
N	S	E	S	E
2000 * (100*100)	1,64	0,82	1,63	0,82
3000 * (100*100)	1,63	0,82	1,65	0,82
4000 * (100*100)	1,64	0,82	1,70	0,85
5000 * (100*100)	1,56	0,78	1,76	0,88

Tabela 2: Prikaz pohitritve in učinkovitosti različnih načinov paralelizacije algoritmov napram sekvenčnemu algoritmu.

A - kreiranje niti skozi vsako generacijo.
B - kreiranje niti zgolj na začetku ter sinhronizacija niti na začetku vsake simulirane generacije.

S povečanjem števila niti se izboljša tudi pohitritev algoritma, vendar pri paraleliziranem algoritmu, kjer se niti ustavarijo zgolj na začetku simulacije ter sinhronizirajo po vsakem delovanju za 4 niti ne dobiva več opaznih pohitritev izvajanja algoritma, mnenja sva da je razlog za tem preveliko število niti ter tako več izgubljenega časa pri komunikaciji oz. sinhronizaciji samih niti.