ELEKTROTEHNIŠKI VESTNIK 84(5): 214–220, 2017
IZVIRNI ZNANSTVENI ČLANEK

Primerjava algoritmov za iskanje metričnih sosedov pri simulacijah skupin živih bitij

Jure Demšar

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana, Slovenija
E-pošta: jure.demsar@fri.uni-lj.si

Povzetek. Računalniške simulacije skupin živih bitij se dandanes uporabljajo tako za proučevanje vedenja bitij, kot tudi za generiranje slikovitih animacij za potrebe računalniških iger in filmov. V vseh primerih je dobrodošlo, da se simulacije izvajajo čim hitreje. Ker na vedenje posameznega agenta v simulaciji najbolj vplivajo preostali bližnji agenti (sosednji agenti), je ključno, da le-te najdemo čim hitreje. Raziskave biologov nakazujejo, da je vedenje opazovanega posameznika v skupini živih bitij pri nekaterih vrstah odvisno od njegovih topoloških sosedov (določeno število najbližjih sosedov), pri drugih pa od njegovih metričnih sosedov (vsi sosedi v določenem radiju). Relevantna literatura nakazuje, da je za iskanje topoloških sosedov najhitrejši algoritem, ki uporablja k-d drevesa. V tem raziskovalnem delu me je predvsem zanimalo, kakšna je situacija pri iskanju metričnih sosedov. Ugotovil sem, da ni metode, ki bi bila najboljša v vseh scenarijih, saj je hitrost iskanja metričnih sosedov pri določenih metodah odvisna tudi od vedenja simuliranih živih bitij. Če se živali zadržujejo v manjših skupinah, je najboljša metoda za iskanje metričnih sosedov razdelitev prostora, če pa se živali zadržujejo v večjih skupinah, je najhitrejša metoda k-d drevo.

Ključne besede: umetno življenje, simuliranje skupin živali, metrični sosedi, algoritmi za iskanje sosedov, optimizacija

A comparison of algorithms for finding metric neighbours in simulations of living beings

Computer simulations of groups of living beings are not useful only for studying the behaviour of beings but also for generating spectacular animations for the purpose of gaming and movie industry. In all cases it is desired for simulations to run as fast as possible. Since the behaviour of a single agent inside the simulations is mostly dependant on other nearby agents (neighbours), it is crucial to have a quick method for finding them. The biology literature suggests that the behaviour of an observed individual depends either on its topological neighbours (k-nearest neighbours) or its metric neighbours (all neighbours inside a certain radius). The relevant literature advocates that the k-d trees are the fastest method when searching for topological neighbours. With the help of this study I was assessing which method would be the best when searching for metric neighbours. I found out that there is no clear winner as the performance of some methods for searching metric neighbours also depends on the behaviour of simulated agents. When agents form smaller groups, the best method is spatial partitioning and when agents coalesce into large groups, it is best to use a k-d tree.

Keywords: artificial life, simulating groups of animals, metric neighbours, neighbour search algorithms, optimization

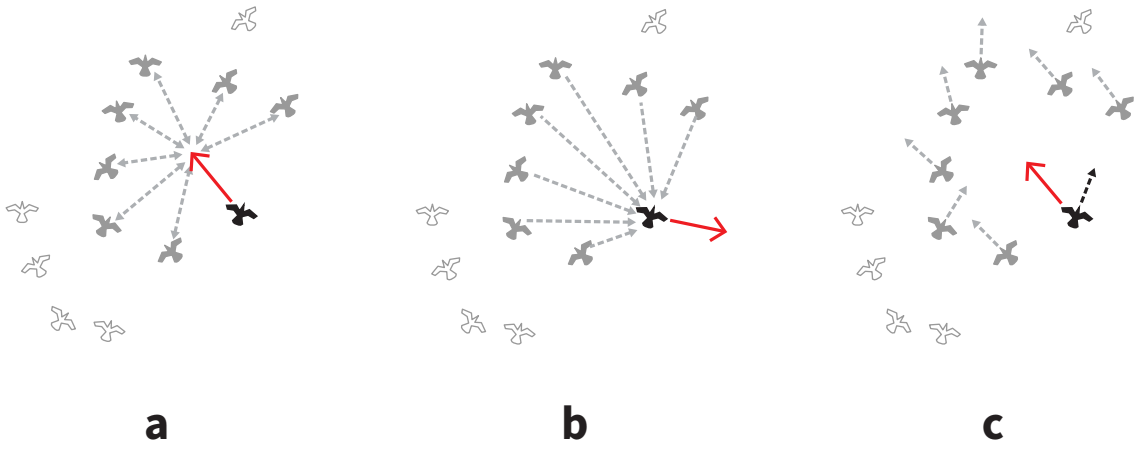
Prejet 27. september, 2017
Odobren 6. december, 2017

1 UVOD

V naravi pogosto opazimo živali, ki se zadržujejo v skupinah. Najbolj tipični primeri so jate ptic in rib, roji žuželk, črede kopitarjev itd. Proučevanje skupin živali že dalj časa ni samo domena biologov, s tem področjem se ukvarjajo tudi znanstveniki iz drugih ved – od fizike in medicine do družbenih ved in računalništva [1], [2], [3], [4].

Računalničarji na tem področju sodelujemo večinoma skozi izdelavo računalniških modelov, ki posnemajo obnašanje skupin živali v naravi. Tovrstni modeli so se izkazali za uporabno metodo pri proučevanju vedenja skupin živali [5], [6], [7], [8], [9], [10], [3], [11], pri simuliranju evolucije skupinskega vedenja [12], [2], [13], [14], [15], [16] in pri izdelavi slikovitih animacij za potrebe računalniških iger in filmov [17], [18], [19], [20]. V vseh primerih je hitrost procesiranja ključnega pomena – pri znanstvenih raziskavah nam višja hitrost omogoča hitrejše pridobivanje rezultatov, medtem ko je pri igrah in filmih visoka hitrost procesiranja po navadi potrebna za izris simulacij skupin živih bitij v realnem času.

Najpogostejši način za računalniško simuliranje skupin živali so modeli, zasnovani na ravni posameznika (angl. *individual-based models*). V tovrstnih modelih je vsaka žival simulirana kot samostojen, avtonomen agent, ki je sposoben zaznavati svet okoli sebe. S pomočjo informacij, pridobljenih skozi zaznavanje sveta, nato



Slika 1: Vizualizacija treh osnovnih teženj: a) kohezije, b) razmika in c) poravnave. Črni agent je opazovani posameznik, sivi agenti pa so sosedi, ki neposredno vplivajo na njegovo vedenje. Beli agenti so zunaj radija zaznave opazovanega posameznika in posledično nimajo neposrednega vpliva na njegovo vedenje.

posamezen agent prilagodi svoje vedenje z namenom, da bi čim bolje zadovoljil svoje težnje (angl. *drives*). Ker so tovrstni modeli zgrajeni na najnižji ravni (na ravni posameznega agenta), pri načrtovanju vedenja simuliranih živih bitij omogočajo veliko fleksibilnost in natančnost [17], [21]. Za visoko natančnostjo pa se skriva tudi največja težava modelov, zasnovanih na ravni posameznika – visoka računska zahtevnost [17]. Zaradi visoke računske zahtevnosti se pogosto zgodi, da načrtovani računalniški model ni sposoben v realnem času simulirati tako velikih skupin živali, kot jih lahko najdemo v naravi. Zelo velike skupine živali v naravi niso redkost, saj velikanske skupine (tisoč in več posameznikov) najdemo pri pticah [22], ribah [23] in kopitarjih [24].

V želji, da bi v realnem času lahko simulirali čim večje skupine živali, se raziskovalci poslužujejo različnih tehnik za pohitritev izvajanja simulacij. Najpogostejša izmed njih je verjetno vzporedno procesiranje [25], [26], [27]. Pri najpogostejši načinih vzporednega procesiranja se za pohitritev uporabljajo grafične kartice, vendar ta način pohitritve razvijalcem ni vedno na voljo – primer so mobilne aplikacije oziroma igre, kjer večina mobilnih naprav ne podpira vzporednega procesiranja na grafičnih karticah. Poleg vzporednega procesiranja obstajajo tudi tehnike modeliranja, ki so sicer hitrejše, vendar tudi precej manj natančne kot modeli, zasnovani na ravni posameznika. Najpogostejša izmed teh tehnik se imenuje modeliranje na principu tokov (angl. *flow-based models*) [28], [17], [29], [21], [20]. Ta tehnika se največkrat uporablja, kadar raziskovalce zanimajo predvsem globalni vzorci, ki se pojavijo na ravni skupine in ne lokalne interakcije med posameznimi člani skupine. Gibanje skupin v tovrstnih modelih spominja na tok tekočine, od koder izhaja tudi ime tehnike. Računsko so tovrstni modeli precej manj zahtevni kot modeli, zasnovani na ravni posameznika, saj je vedenje zasnovano na ravni skupin in ne na ravni posameznih agentov. Posledica modeliranja na višji ravni pa je precej nižja natančnost in s tem tudi dvomljiva biološka relevanca vedenja. Modeli, ki združujejo tehnike obeh svetov, se imenujejo hibridni modeli (angl. *hybrid models*) [17], [30]. Hibridni modeli skušajo združiti natančnost modelov zasnovanih na ravni posameznikov, s hitrostjo modelov, zasnovanih na principu tokov. Rezultat so modeli, ki omogočajo modeliranje razmeroma velikih skupin z dokaj visoko natančnostjo [31], [17], [30]. Tako kot pri modelih na principu tokov je tudi pri hibridnih modelih biološka relevanca dvomljiva.

Ker je v številnih primerih modeliranja visoka biološka relevanca ključnega pomena, smo velikokrat prisiljeni uporabiti modele, zasnovane na ravni posameznika. Največji računski zalogaj je pri tovrstnih modelih preiskovanje okolice. Da lahko agent pravočasno reagira na spremembe v okolici, jo mora ves čas preiskovati. V večini primerov gre pri preiskovanju okolice za iskanje bližnjih sosedov, saj le-ti po navadi vplivajo na vedenje opazovanega posameznika. Več avtorjev [25], [27], [32] je s pomočjo uporabe različnih podatkovnih struktur uspešno pohitrilo postopek iskanja sosedov.

Raziskave biologov nakazujejo, da na obnašanje opazovanega posameznika v skupini vpliva bodisi določeno število najbližjih sosedov (topološki sosedi) [33] bodisi vsi sosedi v določenem radiju (metrični sosedi) [34]. Vermeulen in sod. [32] so pokazali, da pri iskanju topoloških sosedov le-te najhitreje najdemo s pomočjo k-d drevesa [35]. Ker se metode za iskanje metričnih sosedov v določenih primerih razlikujejo od metod za iskanje topoloških sosedov, v tej raziskavi empirično primerjam tri pogoste tehnike iskanja metričnih sosedov –

Tabela 1: Opis in vrednosti uporabljenih parametrov modela.

Parameter	Opis	Vrednost
T	Število korakov	1200
N	Število agentov	100 – 3000
r_r	Radij območja gnezdenja	50 m
w_r	Utež gnezdenja	5
r_s	Radij razmika	1 m
w_s	Utež razmika	10
r_a	Radij poravnave	3 m
w_a	Utež poravnave	3
r_c	Radij zaznavanja/kohezije	5 m, 10 m
w_c	Utež kohezije	3
v_{max}	Najvišja hitrost agentov	20 m/s
m	Masa agentov	1 kg

naivno iskanje, k-d drevesa (angl. *k-d tree*) in razdelitev prostora (angl. *spatial partitioning*).

2 METODE

Za izhodišče pri testiranju hitrosti iskanja metričnih sosedov sem si izbral najbolj znan in razširjen računalniški model za simulacijo letenja ptic v jati BOIDS [19]. To je model, zasnovan na ravni posameznika, pri katerem vsak od simuliranih agentov skuša zadovoljiti tri težnje: razmik (angl. *separation*), poravnavo (angl. *alignment*) in kohezijo (angl. *cohesion*). S pomočjo težnje razmika agenti preprečujejo medsebojne trke in ohranjajo osebni prostor, s pomočjo poravnave usklajujejo hitrost in smer gibanja s svojimi sosedi, s pomočjo kohezije pa tvorijo skupine. Na sliki 1 je grafični prikaz teženj.

V razvitem računalniškem modelu je vsak agent definiran s svojo pozicijo ($\vec{p}_i(t)$, pomeni pozicijo agenta i v nekem trenutku v času) in vektorjem hitrosti ($\vec{v}_i(t)$). Simulacija deluje tako, da vsak simulirani agent v vsakem časovnem koraku (angl. *update step*) prilagodi svoje vedenje glede na stanje sveta okoli njega, da bi zadovoljil svoje težnje. Stanje sveta s stališča opazovanega agenta je definirano skozi njegove metrične sosede – agente, ki se nahajajo v radiju zaznave (r_c) opazovanega agenta. Vsak agent se skuša gibati tako, da bo stanje sveta okoli njega čim bolje zadovoljevalo njegove težnje.

Za zadovoljitev težnje razmika se skuša opazovani agent oddaljiti od vseh sosednjih agentov, ki so mu preblizu (bliže od radija razmika, r_s):

$$\vec{f}_{s,i}(t) = \sum_{j \in N_{s,i}(t)} \hat{o}_{ij}(t) \cdot (r_s^2 - |\vec{o}_{ij}(t)|^2), \qquad (1)$$

kjer je $f_{s,i}(t)$ sila separacije opazovanega agenta i v danem časovnem koraku, $N_{s,i}(t)$ množica vseh i-jevih sosedov, ki so v radiju razmika, in $o_{ij}(t)$:

$$\vec{o}_{ij}(t) = \vec{p}_i(t) - \vec{p}_j(t). \qquad (2)$$

Skozi težnjo poravnave opazovani agent poskuša uskladiti hitrost in smer gibanja z agenti, ki niso preblizu (dalj od radija razmika) in hkrati ne predaleč (bliže od radija poravnave, r_a):

$$\vec{f}_{a,i}(t) = \frac{\sum_{j \in N_{a,i}(t)} \vec{v}_j(t)}{|N_{a,i}(t)|}, \qquad (3)$$

kjer je $f_{s,i}(t)$ sila poravnave in $N_{a,i}(t)$ množica agentov, s katerimi opazovani agent skuša uskladiti svoje gibanje. Skozi težnjo kohezije se opazovani agent približa agentom, ki se mu zdijo predaleč (dalj od r_a):

$$\vec{f}_{c,i}(t) = \frac{\sum_{j \in N_{c,i}(t)} \vec{p}_j(t)}{|N_{c,i}(t)|} - p_i(t), \qquad (4)$$

$f_{c,i}(t)$ pomeni silo približevanja, $N_{c,i}(t)$ pa množico agentov, katerim se opazovani agent poskuša približati. Poleg treh osnovnih teženj, ki jih je definiral Reynolds [19], sem dodal še težnjo po zadrževanju v območju gnezdenja (angl. *roosting area*). Če simulirana ptica zapusti območje gnezdenja, se s pomočjo te težnje vrne "domov":

$$\vec{f}_{r,i}(t) = -\hat{p}_i \cdot (|p_i| - r_r), \qquad (5)$$

kjer $f_{r,i}(t)$ pomeni silo zadrževanja v območju gnezdenja in r_r radij, ki določa velikost območja gnezdenja. Končna sila, ki se uporabi pri izračunu nove pozicije agenta, se izračuna kot:

$$\begin{aligned} \vec{f}_i(t) = \vec{f}_{s,i}(t) \cdot w_s + \vec{f}_{a,i}(t) \cdot w_a + \\ \vec{f}_{c,i}(t) \cdot w_c + \vec{f}_{r,i}(t) \cdot w_r, \end{aligned} \qquad (6)$$

kjer so w_s, w_a, w_c in w_r uteži, ki določajo pomembnost posamezne težnje. Dobljena sila se nato uporabi za izračun nove hitrosti agenta:

$$\vec{v}_i(t) = \vec{v}_i(t - \Delta t) + \frac{\vec{f}_i(t)}{m} \cdot \Delta t, \qquad (7)$$

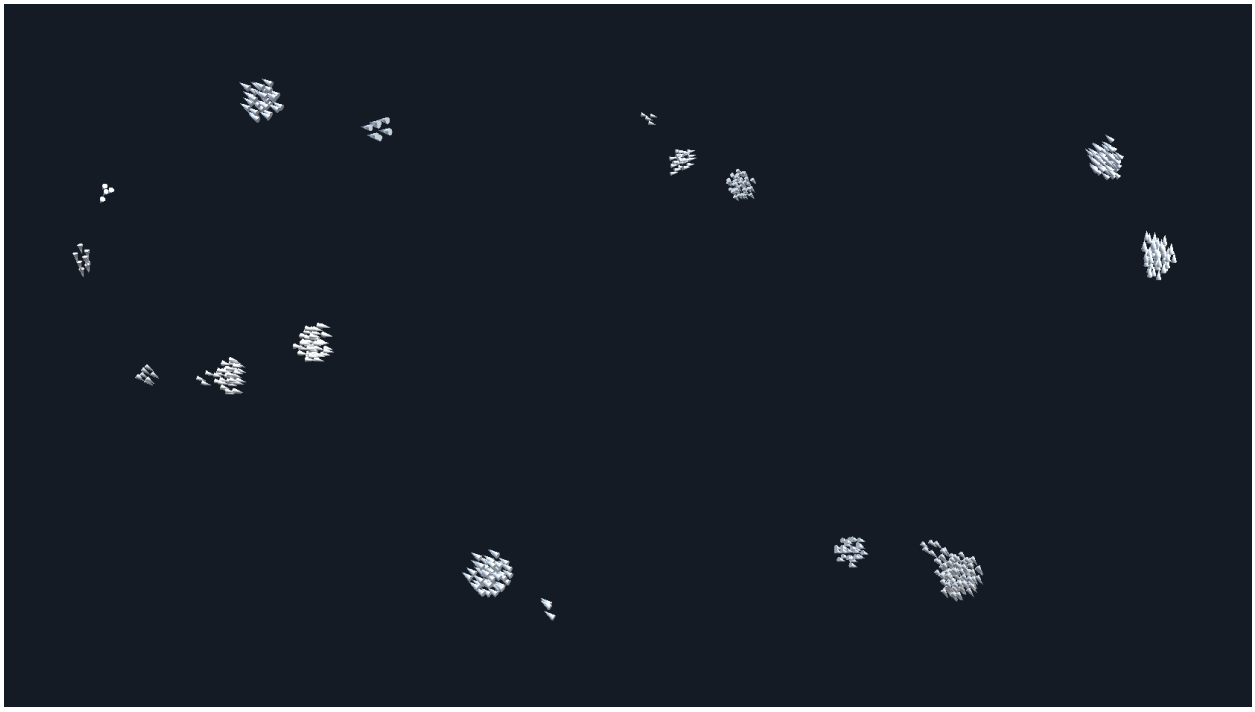
kjer je m masa agenta. Dolžina vektorja hitrosti je navzgor omejena z maksimalno hitrostjo (v_{max}). Nova pozicija agenta tako postane:

$$\vec{p}_i(t + \Delta t) = \vec{p}_i(t) + \vec{v}_i(t) \cdot \Delta t. \qquad (8)$$

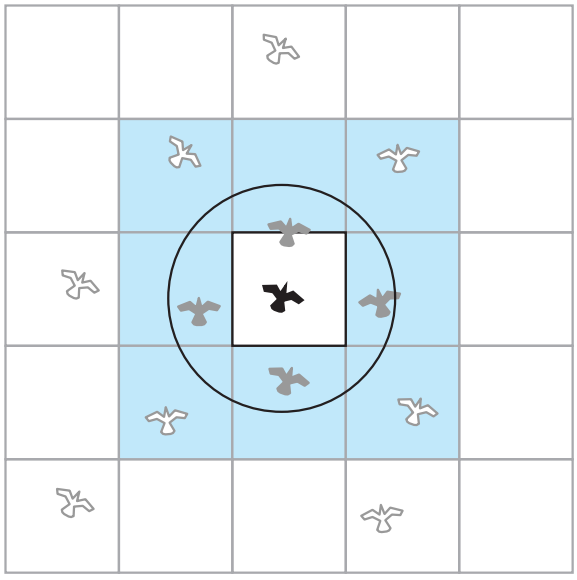
Kratek opis in vrednost uporabljenih parametrov najdemo v tabeli 1.

Poleg same primerjave hitrosti iskanja metričnih sosedov med iskanjem s k-d drevesom in razdelitvijo prostora me je zanimalo tudi, kakšna je pohitritev obeh metod v primerjavi z naivnim pristopom. Pri naivnem pristopu vsak agent v vsakem časovnem koraku izračuna oddaljenost do preostalih $N-1$ agentov ter tako poišče tiste, ki so od njega oddaljeni manj od radija zaznave. Pri naivnem preiskovanju metričnih sosedov imamo torej časovno zahtevnost $O(N^2)$.

K-d drevo je posebna oblika binarnega preiskovalnega drevesa, pri kateri je vsako vozlišče k-dimenzionalna točka [35]. Vsako notranje vozlišče v drevesu razdeli prostor na dva dela s pomočjo hiperravnine. Točke, ki se



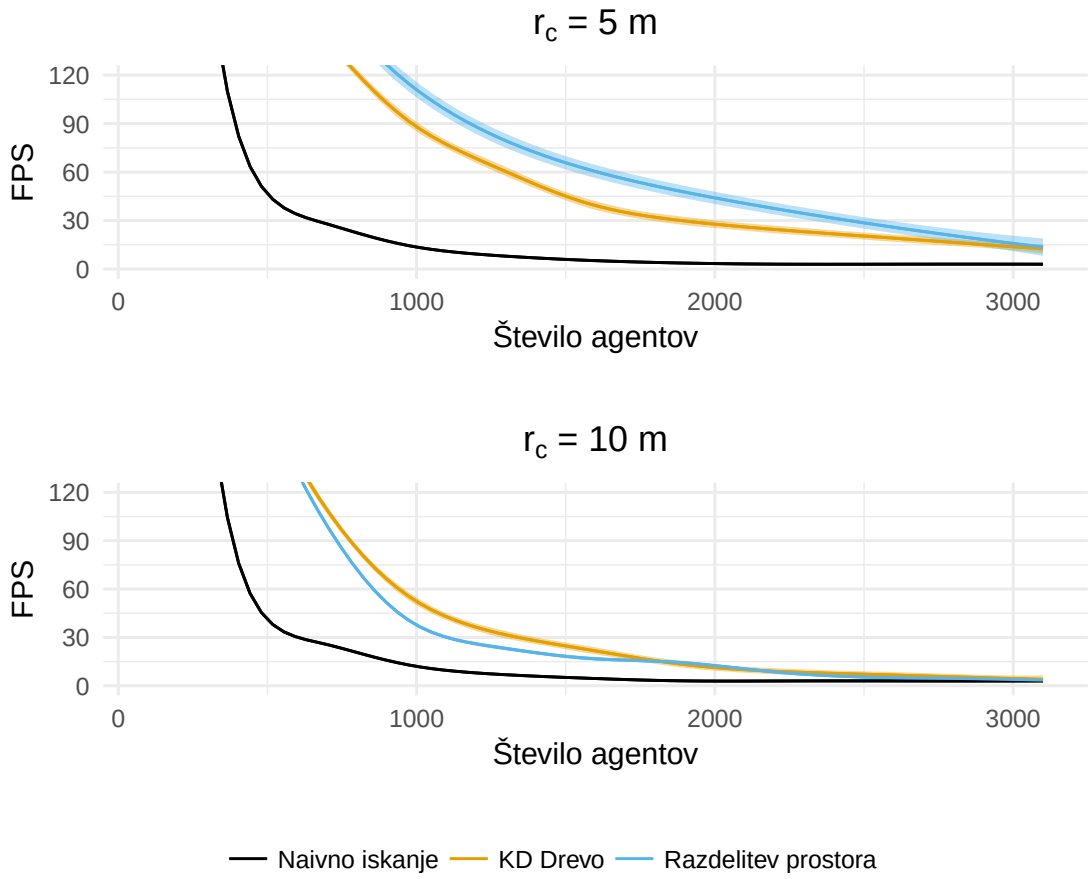
Slika 2: Posnetek zaslona iz ene izmed simulacij. Nastavitev radija zaznave je bila tukaj manjša (r_c = 5 m). Kot vidimo na sliki, se v tem primeru tvori več manjših skupin agentov.



Slika 3: Prikaz iskanja sosedov pri uporabi razdelitve prostora. Agent v črnem je opazovani posameznik, krog s črno obrobo je njegov radij zaznavanja. Pri iskanju sosedov mora opazovani agent preiskati zgolj celico, v kateri se trenutno nahaja (celica s črno obrobo), in sosednje celice (modre celice). Sivi agenti so sosedi, ki so v radiju zaznave in vplivajo na vedenje opazovanega posameznika, medtem ko beli agenti na vedenje opazovanega posameznika ne vplivajo, saj so od njega preveč oddaljeni.

v virtualnem svetu nahajajo na levi strani opazovanega vozlišča, najdemo v levem poddrevesu, tiste, ki se nahajajo desno, pa v desnem poddrevesu. Delitev prostora poteka ciklično po vseh dimenzijah prostora, v katerem se nahajajo točke. Pri treh dimenzijah je tako po navadi v korenu drevesa delitev po x-osi, torej je hiperravnina za delitev pravokotna na x-os. V levem poddrevesu bodo tako točke, ki imajo x manjši od točke v korenu, v desnem pa tiste, ki imajo večji x. Na naslednji ravni se prostor razdeli po naslednji dimenziji (v našem primeru je to y), na naslednji ravni po z, nato zopet po x itd. Pri delitvi prostora je orientacija hiperploskve pravokotna na os, po kateri delimo prostor. Gradnja drevesa ima časovno zahtevnost $O(n \cdot \log n)$, enako časovno zahtevnost ima tudi iskanje metričnih sosedov s pomočjo k-d drevesa [36]. K-d drevo je prvotno namenjeno hranjenju statičnih točk, ker se agenti v našem primeru premikajo, to pomeni, da moramo v vsakem časovni korak najprej zgraditi k-d drevo, nato pa ga uporabiti za iskanje sosedov. Graditev drevesa v vsakem koraku je dodatna računska obremenitev, v nasprotju z naivnim pristopom, kjer so agenti in njihovo stanje po navadi že shranjeni v seznamu, ki ga nato pregledujemo pri iskanju sosedov.

Pri razdelitvi prostora za pohitritev iskanja metričnih sosedov je izkoriščeno dejstvo, da mora vsak agent pri iskanju metričnih sosedov preiskati zgolj bližnjo okolico in ne celotnega virtualnega sveta [27], [32]. Pri razdelitvi prostora se virtualni svet najprej razdeli na manjša območja, v obliki kocke (celice). Stranica kockaste celice je po navadi enaka kot radij, v katerem



Slika 4: Vizualizacija hitrosti izvajanja simulacij v odvisnosti od števila agentov pri dveh različnih velikostih radija zaznavanja. Prikazano je povprečno število sličic na sekundo (FPS) pri določenem številu agentov in 95-odstoten interval zaupanja (obarvan pas okoli črte). Pri manjšem radiju (zgornji graf) se v simulaciji tvori več manjših skupin, pri večjem radiju pa so skupine večje, a jih je manj. Opazimo, da je pri bolj enakomerni porazdelitvi agentov po prostoru (večje število manjših skupin) algoritem, ki išče metrične sosede na podlagi razdelitve prostora, hitrejši kot uporaba k-d drevesa. Kadar pa so agenti zgoščeni v večje skupine, postane uporaba k-d drevesa bolj smiselna kot razdelitev prostora.

iščemo metrične sosede. Tako opazovani agent med iskanjem svoje metrične sosede išče zgolj v celici, v kateri se trenutno nahaja, in v celicah, ki so sosednje trenutni celici. V najslabšem primeru (ko so vsi agenti v eni celici) je časovna zahtevnost enaka kot pri naivnem pristopu, ampak ker do tega scenarija v praksi po navadi ne pride, je pohitritev iskanja sosedov s to tehniko občutna [27], [32]. Princip razdelitve prostora pri dvodimenzionalnem svetu je prikazan na sliki 3.

Primerjavo algoritmov sem izvedel s pomočjo merjenja hitrosti izrisa simulacij. Izris se po navadi meri skozi število slik na sekundo (angl. *frames per second, FPS*), ki jih je simulacija sposobna izrisati pri določenem številu agentov. Model je bil razvit s pomočjo pogona Unity [37]. Simulacije so bila zagnane na računalniku z naslednjo konfiguracijo: procesor Intel Core i7-6700 (3,40 GHz), 32 GB RAM in grafična kartica NVIDIA GeForce GTX 1070. Simulacije so tekle pri resoluciji 4K (3840x2160). Da bi dosegel največje mogoče število slik na sekundo ter da grafične nastavitvne ne bi pretirano vplivale na rezultate, je bila kakovost izrisa najslabša mogoča (brez senčenja in glajenja robov). Pri simulacijah sem spreminjal algoritem za iskanje metričnih sosedov in število agentov v modelu. Vsaka simulacija je tekla 60 sekund – prvih 15 sekund je bilo namenjenih inicializaciji, naslednjih 45 pa merjenju števila slik na sekundo. Algoritme iskanja sosedov sem primerjal skozi povprečno število slik na sekundo, doseženih v teh 45 sekundah.

Pri razdelitvi prostora na hitrost iskanja sosedov vplivata tudi porazdelitev agentov po prostoru in velikost celice pri razdelitvi prostora. Tako razdelitev agentov po prostoru, kot tudi velikost celice, sta v mojem primeru odvisna od radija zaznave/kohezije (r_c). Pri manjših vrednostih radija je razporeditev agentov po prostoru bolj enakomerna, saj se tvori več manjših skupin agentov. Pri večjem radiju se nasprotno tvori manjše število večjih skupin, torej so agenti zgoščeni na manjšem

delu celotnega prostora. Pri testiranju hitrosti iskanja metričnih sosedov sem uporabil dve vrednosti radija zaznave, vsaka od njiju pokrije enega od prej opisanih scenarijev. Slika 2 prikazuje primer ene od simulacij za prvi scenarij – tvori se večje število manjših skupin.

3 Rezultati

Rezultati meritev so prikazani na sliki 4. Pri prvem scenariju (manjši radij zaznave) je bila za iskanje metričnih sosedov najhitrejša metoda razdelitve prostora. Sledilo je iskanje sosedov z uporabo k-d drevesa, pričakovano je bil najpočasnejši naivni pristop. Pri razdelitvi prostora je za hitrost iskanja metričnih sosedov ključna porazdelitev agentov po virtualnem svetu. Do najslabših rezultatov pridemo, če se vsi agenti nahajajo v eni celici. V tem primeru mora vsak agent izračunati razdaljo do vseh drugih agentov ter preveriti, ali je ta razdalja manjša od radija zaznave. Rezultat je enaka hitrost iskanja sosedov kot pri naivni metodi. Če je razporeditev agentov po prostoru bolj enakomerna, mora vsak agent pri iskanju metričnih sosedov izračunati razdaljo zgolj do majhne podmnožice vseh agentov, zato takrat pride to občutljive pohitritve iskanja metričnih sosedov. To lepo prikazuje slika 4 – uporaba razdelitve prostora pri bolj enakomerni razporeditvi agentov (zgornji graf) je precej hitrejša kot pri scenariju, ko se večina agentov zgosti na določenem območju (spodnji graf).

Pri načrtovanju realnočasovnih simulacij težimo k hitrostim izrisa, ki so večje od 60 sličic na sekundo [38]. Pri manjšem radiju zaznave je simulacija, ki uporablja razdelitev prostora, sposobna procesirati s hitrostjo 60 sličic na sekundo pri 1500 agentih, k-d drevo je sposobno procesirati 1300 agentov pri 60 sličicah na sekundo, medtem ko naivna metoda doseže hitrost izrisa več kot 60 sličic na sekundo pri zgolj 400 agentih.

Pri večjem radiju zaznave se je za najboljšo izkazala metoda iskanja metričnih sosedov s pomočjo k-d drevesa, sledila je uporaba razdelitve prostora, na zadnjem mestu pa je bil zopet naivni pristop. K-d drevo je zmožno procesirati 900 agentov s hitrostjo izrisa, višjo od 60 sličic na sekundo, razdelitev prostora lahko procesira 800 agentov, naivna metoda pa zopet zgolj 400 agentov.

4 Sklep

V raziskavi sem primerjal hitrost iskanja metričnih sosedov v računalniških simulacijah živih bitij pri treh različnih metodah. Primerjal sem najpreprostejšo, naivno metodo in dve metodi za pohitritev iskanja – k-d drevo in razdelitev prostora. Obstoječe raziskave [32] nakazujejo, da je za iskanje topoloških sosedov (določeno število najbližjih agentov) nejhitrejša uporaba k-d dreves. Pri iskanju metričnih sosedov pa ta metoda ne da vedno najboljših rezultatov, saj je optimalna metoda pri metričnih sosedih odvisna od vedenja agentov. Če se tvori manjše število večjih skupin agentov, je uporaba k-d drevesa najboljša možnost. V nasprotnem primeru, ko se tvori večje število manjših skupin, pa se je za najprimernejšo izkazala uporaba razdelitve prostora.

Literatura

[1] T. S. Deisboeck and I. D. Couzin, "Collective behavior in cancer cell populations," *BioEssays*, vol. 31, no. 2, pp. 190–197, 2009.

[2] J. Demšar, *Evolution of fuzzy animats in a competitive environment*. PhD thesis, University of Ljubljana, Faculty of Computer and Information Science, 2017.

[3] I. Lebar Bajec and F. H. Heppner, "Organized flight in birds," *Animal Behaviour*, vol. 78, no. 4, pp. 777–789, 2009.

[4] D. J. T. Sumpter, "The principles of collective animal behaviour," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 361, no. 1465, pp. 5–22, 2006.

[5] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective Memory and Spatial Sorting in Animal Groups," *Journal of Theoretical Biology*, vol. 218, no. 1, pp. 1–11, 2002.

[6] J. Demšar and I. Lebar Bajec, "Simulated Predator Attacks on Flocks: A Comparison of Tactics," *Artificial Life*, vol. 20, no. 3, pp. 343–359, 2014.

[7] J. Demšar, C. K. Hemelrijk, H. Hildenbrandt, and I. Lebar Bajec, "Simulating predator attacks on schools: Evolving composite tactics," *Ecological Modelling*, vol. 304, pp. 22–33, 2015.

[8] F. Ginelli, F. Peruani, M.-H. Pillot, H. Chaté, G. Theraulaz, R. Bon, and G. Parisi, "Intermittent collective dynamics emerge from conflicting imperatives in sheep herds," *PNAS*, vol. 112, no. 41, pp. 12729–12734, 2015.

[9] F. H. Heppner and U. Grenander, "A stochastic nonlinear model for coordinated bird flocks," in *The Ubiquity of Chaos* (E. Krasner, ed.), pp. 233–238, AAAS Publications, 1990.

[10] H. Hildenbrandt, C. Carere, and C. K. Hemelrijk, "Self-organized aerial displays of thousands of starlings: A model," *Behavioral Ecology*, vol. 21, no. 6, pp. 1349–1359, 2010.

[11] T. Vicsek, A. Czirok, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel Type of Phase Transition in a System of Self-Driven Particles," *Physical Review Letters*, vol. 75, no. 6, pp. 1226–1229, 1995.

[12] J. Demšar, E. Štrumbelj, and I. Lebar Bajec, "A Balanced Mixture of Antagonistic Pressures Promotes the Evolution of Parallel Movement," *Scientific Reports*, vol. 6, 2016.

[13] J. Demšar and I. Lebar Bajec, "Evolution of Collective Behaviour in an Artificial World Using Linguistic Fuzzy Rule-Based Systems," *PLoS ONE*, vol. 12, no. 1, pp. 1–20, 2017.

[14] A. M. Hein, S. B. Rosenthal, G. I. Hagstrom, A. Berdahl, C. J. Torney, and I. D. Couzin, "The evolution of distributed sensing and collective computation in animal populations," *eLife*, vol. 4, pp. 1–43, 2015.

[15] R. S. Olson, A. Hintze, F. C. Dyer, D. B. Knoester, and C. Adami, "Predator confusion is sufficient to evolve swarming behaviour," *Journal of The Royal Society Interface*, vol. 10, no. 85, 2013.

[16] R. S. Olson, D. B. Knoester, and C. Adami, "Evolution of Swarming Behavior Is Shaped by How Predators Attack," *Artificial Life*, vol. 22, no. 3, pp. 299–318, 2016.

[17] K. Ijaz, S. Sohail, and S. Hashish, "A Survey of Latest Approaches for Crowd Simulation and Modeling using Hybrid Techniques," in *17th UkSim-AMSS International Conference on Modelling and Simulation*, (Cambridge, UK), pp. 111–116, IEEE, 2015.

[18] J. Link, "MASSIVE: Software Engineering for Multiagent Systems," 1999.

[19] C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[20] S. Zhou, D. Chen, W. Cai, L. Luo, M. Y. H. Low, F. Tian, V. S.-H. Tay, D. W. S. Ong, and B. D. Hamilton, "Crowd modeling and simulation technologies," *ACM Transactions on Modeling and Computer Simulation*, vol. 20, no. 4, pp. 1–35, 2010.

[21] M.-L. Xu, H. Jiang, X.-G. Jin, and Z. Deng, "Crowd Simulation and Its Applications: Recent Advances," *Journal of Computer Science and Technology*, vol. 29, no. 5, pp. 799–811, 2014.

[22] A. J. King and D. J. T. Sumpter, "Murmurations," *Current Biology*, vol. 22, no. 4, pp. 112–114, 2012.
[23] D. V. Radakov, *Schooling in the ecology of fish*. New York: Halsted Press, 1973.
[24] S. Creel and J. A. Winnie, "Responses of elk herd size to fine-scale spatial and temporal variation in the risk of predation by wolves," *Animal Behaviour*, vol. 69, no. 5, pp. 1181–1189, 2005.
[25] A. R. Da Silva, W. S. Lages, and L. Chaimowicz, "Boids that See: Using Self-Occlusion for Simulating Large Groups on GPUs," *ACM Computers in Entertainment*, vol. 7, no. 4, p. Article 51, 2009.
[26] A. V. Husselmann and K. A. Hawick, "Simulating species interactions and complex emergence in multiple flocks of Boids with GPUS," *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, no. April 2017, pp. 100–107, 2011.
[27] C. Reynolds, "Big fast crowds on PS3," in *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames - Sandbox '06*, pp. 113–121, 2006.
[28] R. L. Hughes, "The Flow of Human Crowds," *Annual Review of Physical Chemistry*, vol. 35, no. 1, pp. 169–182, 2003.
[29] F. Morini, B. Yersin, J. Maim, and D. Thalman, "Real-Time Scalable Motion Planning for Crowds," in *Proceedings - 2007 International Conference on Cyberworlds, CW'07*, (Hannover), pp. 144–151, IEEE, 2007.
[30] S. I. Park, Y. Cao, and F. Quek, "Large Scale Crowd Simulation Using A Hybrid Agent Model," *Motion in Games*, 2011.
[31] A. Golas, R. Narain, and M. Lin, "Hybrid Long-Range Collision Avoidance for Crowd Simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 7, pp. 1022–1034, 2014.
[32] J. L. Vermeulen, A. Hillebrand, and R. Geraerts, "A comparative study of k-nearest neighbour techniques in crowd simulation," *Computer Animation and Virtual Worlds*, 2017.
[33] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, V. Lecomte, and A. Orlandi, "Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study," *Pnas*, vol. 105, no. 4, pp. 1232–1237, 2008.
[34] D. J. Evangelista, D. D. Ray, S. K. Raja, and T. L. Hedrick, "Three-dimensional trajectories and network analyses of group behaviour within chimney swift flocks during approaches to the roost," *Proc. R. Soc. B*, vol. 284, no. 1849, 2017.
[35] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
[36] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, third ed., 2009.
[37] Unity Technologies, "Unity - Game Engine," 2017.
[38] D. Andreev, "Real-time Frame Rate Up-conversion for Video Games: Or How to Get from 30 to 60 Fps for Free," in *ACM SIGGRAPH 2010 Talks*, p. 1, ACM, 2010.

Jure Demšar je diplomiral (2011) in doktoriral (2017) na Fakulteti za računalništvo in informatiko, kjer je trenutno zaposlen kot asistent in raziskovalec. Raziskovalno se ukvarja z modeliranjem in simuliranjem skupinskega vedenja, razvojem iger in računalniškim modeliranjem možganov.