Analiza ponašanja agenta nasumičnog šetanja u imitaciji kretanja vinske mušice u 2D prostoru

Benington, Mihailo

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Mihailo Benington

Analiza ponašanja agenta nasumičnog šetanja u imitaciji kretanja vinske mušice u 2D prostoru

Završni rad

Mentor: prof. dr. sc. Ana Meštrović

Rijeka, kolovoz 2022.

Sadržaj

Sažetak

U ovom radu opisivat će se pojam kompleksnih sustava i nasumičnih šetnji, kao i njihova uloga u prostornim (i drugim) simulacijama pojava u stvarnom svijetu. Opisat će se sustav razvijen za generiranje simulacija šetnji vinskih mušica pomoću nasumičnih šetača na osnovu podataka dobivenih promatranjem vinskih mušica u stvarnom sustavu. Simulirani podatci usporedit će se sa stvarnima i analizirati na osnovu mjera kompleksnih mreža koje su opisane u nastavku.

Ključne riječi: modeli; agenti; šetnje; nasumične šetnje; kompleksni sustavi; prostorna simulacija

1 Uvod

Potražnja za simulacijama pomoću modela i njihova primjena u stvarnom svijetu može se naći u raznim oblastima, uključujući vojne simulacije, logistiku, transportaciju, distribuciju, medicinske simulacije (npr. simuliranje stanica raka), utjecaj klimatskih promjena i planiranje sprječavanja istih, planiranje poslovnih procesa i financijsku domenu (npr, simuliranje kretanja vrijednosti dionica/kriptovaluta).
U znanstvenom istraživanju simulacijski modeli sve se češće koriste kao heuristički alat učenja kako bi se dobilo bolje razumijevanje specifičnog sustava. Prostorni simulacijski modeli su relativno nedavni dodatak znanstvenom repertoaru koji se može gledati kao svojevrsni *virtualni laboratorij* za testiranje hipoteza.
Jedna od najvećih prednosti korištenja simulacija zasnovanih na modelima je ušteda resursa koji bi inače bili potrebni za donošenje zaključaka i izgradnje definicija kompleksnih sustava koji se promatraju.
U nastavku ovog rada definirat će se pojmovi nasumične šetnje, modela, agenta, kompleksnih sustava i proučiti implementacija istih na osnovu podataka sakupljenih promatranjem ponašanja vinskih mušica.
Opisat će se implementacija sustava za simuliranje kretanja vinskih mušica u dvodimenzionalnom prostoru u programskom jeziku Python koristeći podatke dobivene promatranjem stvarnih sustava vinskih mušica i bilježenje njihovih interakcija.
Jedinkama se u simulacijama zadaju smjer kretanja i duljine koraka iz čega se pokušava dobiti grupno socijalno ponašanje koje se provjerava mjerama nad kompleksnim mrežama.
Cilj simulacija je modeliranje kretanja i interagiranja mušica na osnovu postojećih podataka. Za uspoređivanje podataka koristit će se mreže (grafovi) i mjere nad mrežama. U daljnjem tekstu će se za vrh/čvor grafa koristiti izraz mušica, a za brid/vezu izraz interakcija.

2 Teorijska pozadina

2.1 Analiza ponašanja vinskih mušica

Velik broj različitih vrsta životinja se u prošlosti koristio za testiranje nad životinjama uključujući miševe, mušice, majmune i druge. Posebno koristan organizam za proučavanje je obična vinska mušica - *Drosophila melanogaster*. Ova mušica je ubrzo postala glavni model beskičmenjaka koja se koristi u razvojnoj genetici, biologiji i drugim disciplinama, a koristi se već više od stoljeća [1].
Mnoge različite karakteristike vinske mušice ju čine idealnim modelom za promatranje. Ogromna prednost korištenja vinskih mušica je manjak etičkih problema koji se javljaju kod testiranja nekih drugih, kompleksnijih životinja, kao što su, na primjer, majmuni ili neki drugi sisavci.
Kratki životni vijek vinske mušice omogućava uzgoj velikog broja mušica u kratkom vremenskom roku. Zametak se pojavljuje 24 sata od oplodnje jajeta, nakon čega se, nakon tri stadija razvoja, pojavljuje odrasla mušica. Razvoj odrasle jedinke traje samo 10 dana nakon oplodnje. Ženska mušica može proizvesti 1500 jaja u svom životnom vijeku, čime se uspostavlja stalan dotok novih mušica potrebnih za daljnje proučavanje.
Još jedna prednost vinskih mušica je njihova veličina; vinske mušice su vrlo male i, samim tim, vrlo lako održive. Zbog njihove veličine i minimalnih drugih zahtjeva, mnoge mušice se mogu odgajati i testirati unutar jako malih laboratorija koji nemaju ograničeno vrijeme pristupa, prostor ili financiranje.
Genetički faktori su još jedna od stvari koje vinsku mušicu čine idealnim organizmom za proučavanje. Mušice imaju samo četiri para kromosoma (u usporedbi sa ljudskih 23 para). Ova genetska jednostavnost je jedan od razloga zbog kojih su prvotno bile korištene u proučavanju genetike; njihovi geni se vrlo lako mapiraju u svrhu istraživanja genetskog nasljeđivanja. Cijeli genom vinske mušice je sekvenciran i mapiran na sličan način kao ljudski genom (oko 60% sličnosti) dok veličinom iznosi samo 5% ljudskog genoma [2].
vinska mušica ima anatomske karakteristike (poput krila i očiju) koje omogućuju jednostavnu karakteristiku, odnosno omogućuju lako identificiranje jedinke na osnovu istih. Ponašanja kao što su prehrana, parenje, spavanje, agresija i traženje hrane koja se mogu promatrati kod ljudi se također pronalaze kod vinskih mušica, zbog čega se vinske mušice koriste i u svrhu proučavanju socijalnih interakcija i utjecaja genetskog nasljeđivanja na socijalne karakteristike [2], [3].
Jedna od aktualnih tema kod proučavanja vinskih mušica je **utjecaj brojnih vanjskih faktora** na njihovo ponašanje, uključujući izolaciju, metamfetamin i kokain [4], [5], [6]. Fakultet biotehnologije i razvoja lijekova u suradnji sa Fakultetom informatike i digitalnih tehnologija napisao je dva rada koji se bave analizom ponašanja uzoraka mušica koje nisu tretirane psihostimulansima, uzoraka koji jesu tretirani i razlikama u dobivenim uzorcima [5], [6]. Ovaj rad je dio navedenih istraživanja.

2.2 Kompleksni sustavi

Kompleksni sustavi sastoje se od mnoštva komponenti koje međusobno interagiraju. Mnoštvo je primjera kompleksnih sustava koji su nužni za funkcioniranje modernog društva (živi organizmi, ljudski mozak, računala, operacijski sustavi i sl.), štoviše, većina objekata i entiteta iz naše okoline može se svrstati pod kompleksne sustave bez gubitka općenitosti.
Kompleksni sustavi imaju distinktivna svojstva koja proizlaze iz međuodnosa njihovih komponenti kao što su **nelinearnost**, **emergentnost** (engl. *emergence*), **prilagođavanje** i **povratne petlje** (engl. *feedback loops*).
S obzirom da se kompleksni sustavi pojavljuju u širokom spektru područja ljudskog interesa, zajednička svojstva različitih sustava postala su temom nezavisnog područja istraživanja. U većini slučajeva korisno je predstaviti takve sustave kao mrežu (graf) gdje se komponente sustava predstavljaju čvorovima, a njihove interakcije vezama (bridovima).
Izraz *kompleksni sustavi* često se odnosi na proučavanje kompleksnih sustava, što je znanstveni pristup koji istražuje kako odnosi između komponenti sustava rezultiraju kolektivnim ponašanjem i formiranjem odnosa sustava sa okolinom [7]. Proučavanje kompleksnih sustava bazira se na kolektivnom ponašanju kao temelju proučavanja; zbog ovoga se kompleksni sustavi mogu interpretirati kao alternativna paradigma redukcionizmu, paradigmi koja objašnjava sustave preko njihovih komponenti i njihovih individualnih interakcija [8].
Kao interdisciplinarno područje, teorija kompleksnih sustava izvodi svoje zaključke iz različitih područja, kao što su proučavanje samoorganizacije i kritičkih fenomena iz biologije i kemije, kaosa iz matematike, prilagođavanja iz biologije i mnogih drugih. Iz ovog razloga se kompleksni sustavi često koriste kao općeniti termin koji obuhvaća znanstveni pristup problemima u mnogim šarenim disciplinama kao što su statistička fizika, teorija informacija, nelinearna dinamika, antropologija, računarska znanost i druge.
Adaptacija ili prilagodba je svojstvo entiteta da mijenja svoje ponašanje ovisno o okolini i kontekstu u kojemu se nalazi [9].
Kompleksni adaptivni sustavi su posebna vrsta sustava koji imaju kapacitet učenja i prilagodbe ponašanja na osnovu prikupljenih ili proslijeđenih podataka i prethodnih iskustava (drugim riječima, kapacitet za **empiriju**). Primjeri ovakvih sustava su burza, kolonije insekata, imunološki sustav i bilo koja ljudska organizacija ili poslovni model (s rijetkim izuzetcima).
Nelinearni sustavi su sustavi kod kojih promjena u izlazu iz sustava nije proporcionalna ulazu u sustav [11].
Nelinerani sustavi su od interesa raznim znanstvenim disciplinama, s obzirom da je većina sustava nelinearno po prirodi [9]. Primjer nelinearnog sustava je upravo simulacija kretanja mušica u dvodimenzionalnom prostoru, s obzirom da se ista mušica u istom vremenskom razdoblju može kretati na beskonačno mnogo načina ovisno o najmanjim faktorima na koje se ne može utjecati (početna točka upuhivanja mušice, vlaga u zraku, količina prašine na podu arene i slično).
Emergentnost (engl. *emergence*) je pojava u kojem kompleksni sustav ima svojstva ili ponašanja koja njegove komponente samostalno ne posjeduju. Primjeri kompleksnih sustava sa svojstvom emergentnosti u doslovnom smislu se često nalaze u fizici za opisivanje fenomena

koji su prisutni na makroskopskim razinama, ali ne na mikroskopskim, iako se makroskopski sustavi mogu interpretirati kao skup mikroskopskih sustava. Neki od primjera ovakvih sustava bili bi mozak, organ ili organizam, skupina radikalno orijentiranih pojedinaca, terorističke organizacije, skupina ljudi koja protestira protiv normalnog oblačenja žena i slično [12].

2.3 Modeli bazirani na agentima

Modeliranje bazirano na **agentima** (engl. *agent-based modeling*, nadalje ABM) je ukorjenjeno u teoriji kompleksnih sustava. Cilj agentnog modeliranja je modeliranje sustavnih uzoraka koji proizlaze iz međusobnih interakcija individualnih agenata sa svojim susjednim agentima i lokalnim okolišem [14]. Ove interakcije su određene pravilima, primjerice dvije mušice u areni su u interakciji samo ako su im međusobne udaljenosti manje od proizvoljne ϵ vrijednosti i traju dulje od n trenutaka. Implementacija ovih modela zbog toga zahtijeva određenu količinu vještina u programiranju kako bi se pravila za određivanja interakcija implementirala na ispravan i **praktičan** način.

2.3.1 Motivacija za uvođenje modela baziranih na agentima

Svrha ABM je objašnjenje sustavnih svojstava pomoću ponašanja međusobno isprepletenih pojedinaca. Životinja, na primjer, nije svjesna stope nataliteta na razini svoje populacije. Ona će se uspiješno reproducirati ako nađe pogodnog partnera, ako ima dovoljno hrane u blizi staništa i slično. Ponašanje populacije životinja "izranja" iz ponašanja jedinki tokom njihovog životnog ciklusa i njihovih interakcija s drugim jedinkama iste populacije (ili drugih jedinki druge populacije, npr. grabežljvaca unutar istog sustava). Svojstva sustava kao što su veličina populacija, stopa nataliteta i mortaliteta su rezultati simulacije, radije nego ulazni podatci (iako je i ovo moguće ovisno o zahtijevima simulacije). ABM je u biologiji i medicini uhvatilo zamaha upravo zbog jakih sličnosti sa stvarnim živućim sustavima, gdje su glavne jedinke sustava jedinke iz carstva biljaka ili životinja .

2.3.2 Agenti

Agent u ABM je diskretan i autonomni entitet koji se samostalno ponaša prilikom reagiranja na svoj okoliš. U najjednostavnim slučajevima, agenti mogu biti biljke, a u kompleksnijim životinje ili ljudi koji streme ka nekom cilju, imaju sposobnost pamćenja, učenja i prilagodbe.
Iz perspektive računalne znanosti, agenti su osnovni element za građenje ABM. Softverski su najčešće implementirani u objektno-orijentiranoj paradigmi u kojoj se diskretni agenti (objekti) opisuju sa svojim potencijalnim ponašanjem (metodama ili funkcijama). Objektno-orijentirana paradigma je izrazito prigodna za ABM zbog jednostavno mogućeg opisivanja pojedinačnih agenata kao instanci specifičnog tipa (klase). Ovo znači da će svaki agent imati iste atribute (dob, spol, veličinu) i isto ponašanje (kretanje, glasanje, komunikaciju), ali će vrijednosti tih atributa (i samim tim, ponašanja) u stvarnosti varirati.

2.3.3 Ponašanje

Ponašanje agenata u ABM je skup unaprijed definiranih pravila pomoću kojih agent interagira s okolicom ili drugim agentima (primjerice agent koji modelira pješaka na ulici će početi razgovarati s drugim agentom ako je taj agent u skupu poznanika prvog agenta, ako mu je dovoljno blizu i ako ga prvi agent vidi/primijeti).
Ponašanje se uglavnom izražava kao skup *if-then* pravila koja dirigiraju okolnosti pod kojima određeni agent donosi odluke i/ili odabire ciljeve. Pravila ponašanja se mogu derivirati iz teorije ili iz empirijskih podataka (kao što je slučaj u modelu opisanom u ovom radu).

2.4 Osnovni pojmovi iz teorije grafova

Teorija koja opisuje i analizira kompleksne mreže svoje temelje ima u teoriji grafova i statističkoj analizi. Kompleksnu mrežu opisujemo pomoću grafova, tako da podatke prikazujemo vrhovima (koje u kontekstu mreža možemo zvati čvorovima), a veze između njih bridovima. Ovisno o svojstvima mreže, taj graf može biti usmjeren ili neusmjeren, težinski ili bez težina. Radi lakšeg razumijevanja u ovom poglavlju definiramo osnovne pojmove teorije grafova koji se koriste u nastavku rada.

Definicija 1.1 *Graf* G je uređena trojka $G = (V(G),\ E(G),\ \psi_G)$ koja se sastoji od neprazznog skupa $V(G)$ čiji su elementi vrhovi od G, skupa $E(G)$ disjunktnog s $V(G)$ čiji su elementi bridovi od G i funkcije incidencije ψ_G koja svakom bridu pridružuje par (ne nužno različitih) vrhova od G.
U slučaju da je taj par vrhova neuređen, onda govorimo o *neusmjerenom grafu* G.
U suprotnom, ako svakom bridu pridružimo uređeni par vrhova od G, onda govorimo o *usmjerenom grafu (digrafu)*.
Ukoliko svakom bridu $e \in E(G)$ pridružimo težinu $w(e) \in \mathbb{R},\ w(e) \geq 0$, tada graf G s funkcijom w nazivamo *težinskim grafom*.
Graf koji ima samo jedan vrh se naziva *trivijalan graf*.

Definicija 1.2 Kažemo da su krajevi u i v brida e *incidentni* s bridom e, i obratno, brid e je incidentan sa svojim krajevima.

Definicija 1.3 Za dva vrha koji su incidentni s istim bridom kažemo da su *susjedni*.

Definicija 1.4 Brid koji je incidentan samo s jednim vrhom zove se *petlja* te kažemo da je graf *jednostavan* ako nema petlje ni višestrukih bridova (tj. nikoja dva brida ne spajaju isti par vrhova).

Definicija 1.5 Jednostavan graf u kojemu je svaki par vrhova susjedan zove se *potpun graf* i označava s K_n, gdje je n broj vrhova tog grafa.

Definicija 1.6 Graf G' je *podgraf* grafa G ako vrijedi $V(G') \subseteq V(G)$, $E(G') \subseteq E(G)$ i $\psi_{G'}$ je restrikcija od ψ_G na $E(G')$.

Definicija 1.7 Relacija povezanosti je relacija ekvivalencije na skupu $V(G)$. Neka su $V1, V2, ..., V_\omega$ klase ekvivalencije. Podgrafovi $G[V1], G[V2], ..., G[V_\omega]$ su komponente od G. Graf G je *povezan* ako je $\omega = 1$. U suprotnome, graf G je *nepovezan* s $2 \leq \omega$ komponenti.

Definicija 1.8 Neka je G proizvoljan graf sa skupom vrhova $V(G) = \{v_1, v_2, ..., v_n\}$. Matrica susjedstva $A(G)$ grafa G je kvadratna $n \times n$ matrica s elementima a_{ij}, $i, j \in \{1, 2, ..., n\}$, pri čemu je a_{ij} broj bridova koji spajaju vrhove v_i i v_j.

Definicija 1.9 Neka je G graf i $v \in V(G)$. *Stupanj* ili *valencija* vrha v je broj bridova od G incidentnih s v pri čemu svaku petlju brojimo dvaput.
Ukoliko je G usmjeren graf, definiramo *instupanj* (ulazni stupanj) i *outstupanj* (izlazni stupanj) vrha v kao broj lukova s krajem, odnosno početkom u v.
Ukoliko je G težinski graf, tada definiramo *težinski stupanj* vrha v kao zbroj težina bridova incidentnih s vrhom v.

Definicija 1.10 *Šetnja* u grafu G je konačan niz $w = v_o e_1 v_1 e_2 ... e_k v_k$, gdje su
$v_0, v_1, ..., v_k \in V(G),\ e_1, e_2, ..., e_k \in E(G)$ i $e_i = v_{i-1}v_i,\ i = 1, ..., k$. Kažemo da je w šetnja od v_0 do v_k, odnosno (v_0, v_k)-šetnja.

Šetnja je *zatvorena* ako ima pozitivnu duljinu i ako se početak i kraj podudaraju.

Definicija 1.11 Zatvorena šetnja kod koje su svi vrhovi, osim početnog i krajnjeg, međusobno različiti naziva se *ciklus*.

Definicija 1.12 Šetnju w u kojoj su svi bridovi i svi vrhovi međusobno različiti nazivamo *put*.

Definicija 1.13 Neka je $S \subseteq V(G)$. Skup svih vrhova iz $V(G)$ koji su susjedni s barem jednim vrhom iz S nazivamo *susjedstvo* (ili skup susjeda) od S i označavamo s $N_G(S)$ ili $N(S)$.
Posebno, ako je $S = \{v\}$ tada govorimo o susjedstvu vrha $v \in V(G)$ i pišemo $N_G(v)$.

2.5 Nasumične šetnje

Nasumična šetnja je nasumični proces u matematičkom prostoru. Ona opisuje put koji se sastoji od uzastopnih nasumičnih koraka u matematičkom prostoru. Prvi put ju je uveo Pearson 1905. godine [10]. Spitzer je dao kompletan pregled nasumičnih šetnji za matematička istraživanja i jasno je prezentirao njihove matematičke principe. Nasumične šetnje koriste se za analizu i simulaciju nasumičnosti objekata i računanja korelacije između istih, što je izuzetno korisno u rješavanju mnogih praktičnih problema.
U matematičkom prostoru, jednostavni model nasumične šetnje je nasumična šetnja na regular-

noj rešetci[1][17], u kojoj jedna točka može skočiti na drugu poziciju u svakom koraku ovisno o određenoj distribuciji vjerojatnosti.
Nasumična šetnja poznata je kao nasumični proces. Opisuje put koji se sastoji od niza uzastopnih nasumičnih koraka na nekom matematičkom prostoru, koji se može označiti sa $\zeta_t,\ t = 0, 1, 2, ...$, gdje je ζ_t nasumična varijabla koja opisuje poziciju nasumične šetnje nakon t koraka. Ovaj niz se može tumačiti i kao posebna kategorija **Markovljevih lanaca**[2]. U početnom stanju nasumične šetnje, pozicija ζ_0 može biti fiksna ili derivirana iz neke inicijalne distribucije P_0. Možemo predstaviti distribuciju pozicije nakon t koraka na sljedeći način:

$$P_t(i) = Pr(\zeta_t = i),$$

gdje je $P_t(i)$ vjerojatnost da nasumična šetnja posjeti poziciju i nakon t koraka. Ako se šetnja nađe na poziciji i nakon t koraka, vjerojatnost tranzicije jednog koraka odnosi se na vjerojatnost da se nasumična šetnja može naći na poziciji j nakon sljedećeg koraka. Označuje se kao p_{ij} i računa se na sljedeći način:

$$p_{ij} = Pr(\zeta_{t+1} = j|\zeta_t = i).$$

Nadalje, vjerojatnost tranzicije t koraka računa se kao:

$$p_{ij}^{(t)} = Pr(\zeta_t = j|\zeta_0 = i).$$

Iz perspektive predstavljanja grafovima, neka je $G = (V, E)$ povezani graf, gdje je V skup vrhova i E skup bridova. Matrica susjedstva grafa G označava se kao $A \in \mathbb{R}^{n\times n}$, gdje je n broj čvorova u g. A_{ij} označava težinu brida između čvorova i i j. Tada se vjerojatnost tranzicije jednog koraka od čvora i do čvora j definira na sljedeći način:

$$p_{ij} = \frac{A_{ij}}{\sum_{j\in V} A_{ij}}.$$

2.6 Mjere u mrežama

Heterogenost stupnja (d_{het}) predstavlja raznolikost u stupnjevima čvora i raznolikost u strukturi mreže. Računa se kao omjer standardne devijacije i prosječnog stupnja. Heterogenost stupnja ima izravnu korespondenciju s entropijom složene mreže koja je karakterizirana standardnom Shannonovom mjerom informacija.[3] U kontekstu analize mreže vinskih mušica heterogenost stupnja predstavlja razlike u broju interakcija koje mušice imaju, iz čega se mogu dovoditi zaključci o socijalnim značajkama (ekstrovertiranosti, introvertiranosti i slično).
Asortativnost (r) je mjera sklonosti povezivanja čvorova s drugim sličnim čvorovima. Mreža pokazuje asortativno miješanje po stupnju ako čvorovi teže ka povezivanju s drugim čvorovima sličnog stupnja. Ova vrijednost se računa kao Pearsonova korelacija:

[1] Definicija rešetke dostupna je na poveznici
[2] Detaljniji opis Markovljevih lanaca dostupan je na poveznici
[3] Više o Shannovoj mjeri informacije može se pročitati na poveznici

$$r = \frac{\sum_{jk} jk(e_{jk} - q_j q_k)}{\sigma_q^2}, \tag{1}$$

gdje je e_{jk} zajednička distribucija vjerojatnosti prekomjernih stupnjeva [28] dvaju čvorova na svakom kraju nasumično odabranog brida (interakcije). Ovdje su $q_j q_k$ i σ_q^2 očekivana vrijednost ili srednja vrijednost i standardna devijacija distribucije prekomjernog stupnja. U kontekstu analize mreže vinskih mušica asortativnost opisuje koliko su mušice voljne za povezivanje sa drugim mušicama ovisno o tome koliko su druge mušice socijalne (odnosno s koliko su drugih mušica već imale interakciju).

Centralnost blizine (cc_i) čvora odražava koliko je čvor blizu svim drugim čvorovima u mreži i izračunava se kao prosječna duljina najkraćeg puta od čvora do svakog drugog čvora u mreži. Najkraći put između dvaju čvorova je najmanji broj bridova potrebnih za povezivanje tih dvaju čvorova. U kontekstu mreže, čvor s višom vrijednošću centralnosti blizine ima bliske interakcije sa svim ostalim čvorovima u mreži. Za centralnost blizine neka je d_{ij} najkraći put između čvorova i i j. Normalizirana centralnost blizine čvora i zadana je izrazom:

$$cc_i = \frac{N-1}{\sum_{i \neq j} d_{ij}}. \tag{2}$$

U kontekstu analize mreže vinskih mušica centralnost blizine određuje koliko su mušice zbijene ili raštrkane.

Napomena: Centralnost blizine u ovom radu se ponaša nešto drukčije nego što je objašnjeno u gornjem odlomku. S obzirom da težine bridova između čvorova mreža u ovom radu izražavaju broj i duljinu interakcija koje su definirane kao proizvoljno mala udaljenost između mušica, težina brida je obrnuto proporcionalna stvarnoj euklidskoj udaljenosti. Iz ovog razloga dobivene mjere centralnosti blizine se ponašaju na sljedeći način: što je mjera viša, čvorovi su udaljeniji i obrnuto.

Centralnost posredovanja čvora je količina najkraćih puteva između svih parova čvorova u mreži koji sadrže trenutni čvor. Čvor s višom vrijednošću centralnosti posredovanja ima interakcije s različitim zajednicama čvorova u mreži. Neka je σ_{jk} broj najkraćih puteva od čvora j do čvora k i neka je $\sigma_{jk}(i)$ broj tih puteva koji prolaze kroz čvor i. Normalizirana centralnost posredovanja čvora i zadana je sljedećim izrazom:

$$bc_i = \frac{\sum_{i \neq j \neq k} \frac{\sigma_{jk}(i)}{\sigma_{jk}}}{(N-1)(N-2)}. \tag{3}$$

U kontekstu analize mreže vinskih mušica centralnost posredovanja se tumači kao tendencija mušica da se podijele u zajednice.

Zajednice su skupine međusobno gusto povezanih čvorova unutar mreže. Čvorovi u zajednici imaju veći broj bridova međusobno nego s ostalim čvorovima u mreži. Za otkrivanje zajednica koristi se nekoliko algoritama kao što su hijerarhijsko grupiranje, Girvan-Newmanov algoritam, metoda minimalnog presjeka i drugi. Jedan od najučinkovitijih je Louvain algoritam [27]. Louvain algoritam temelji se na metodi pohlepne optimizacije koja optimizira modularnost par-

ticija mreže.

Lokalni koeficijent grupiranja (C_i) čvora mjeri koliko su dobro susjedi međusobno povezani i kvantificira postaju li klika, tj. podgraf u kojemu su svi čvorovi međusobno povezani. Lokalni koeficijent grupiranja izračunava se kao omjer bridova između čvorova unutar njihovog susjedstva i broja bridova koji bi mogli postojati među njima. Mreže iz stvarnog svijeta (posebno društvene mreže) u prosjeku imaju veći koeficijent grupiranja od slučajnih mreža (pri usporedbi mreža iste veličine). Koeficijent grupiranja čvora i definiran je sljedećim izrazom:

$$C_i = \frac{e_{ij}}{k_i(k_i - 1)}, \tag{4}$$

gdje e_{ij} predstavlja broj parova susjednih čvorova koji su povezani. Ova mjera u kontekstu analiziranja mreže vinskih mušica zajedno s centralnošću posredovanja diktira tendenciju povezivanja mušica u zajednice.

Newtonova modularnost (Q) opisuje kvalitetu particioniranja mreže na zajednice. Vrijednost modularnosti je u rasponu $[-0.5, 1]$. Mreže s višim vrijednostima imaju guste veze između čvorova unutar zajednice, ali rijetke veze između čvorova u različitim zajednicama. U kontekstu ovog rada težnja mušica da se podijele u zajednice i ostanu u njima je proporcionalna Newtonovoj modularnosti. Neka je e_{ij} udio interakcija u mreži koji povezuje čvorove u grupi i s onima u grupi j i neka je $a_i = \sum_j e_{ij}$. Tada se modularnost može izračunati pomoću sljedećeg izraza:

$$Q = \sum_{i=1}^{N} (e_{ii} - a_i^2). \tag{5}$$

3 Metoda

Simuliranje kretanja (nadalje **šetanje**) mušica implementirano je pomoću nasumičnih šetača; nasumični šetač je agent sa svojstvima smjera kretanja, trenutne pozicije i svih prošlih pozicija. Prilikom kreiranja šetača, trenutna pozicija i smjer kretanja se nasumično kreiraju. U svakom trenutku uzorkovanja (24 puta u sekundi, 20 minuta) šetač uzima nasumični smjer koji ne smije biti više od 45° lijevo ili desno od trenutnog smjera (odnosno $\frac{\pi}{4}$ u pozitivnom ili negativnom smjeru), drugim riječima, ako je trenutni smjer označen sa α, novi smjer se generira iz otvorenog intervala $\langle \alpha - \frac{\pi}{4}, \alpha + \frac{\pi}{4} \rangle$. Duljina koraka koji šetač napravi uzima se nasumično iz liste koja sadrži distribuciju duljina koraka iz stvarnih eksperimenata.
Nakon svih šetnji u simulaciji bilježe se interakcije. Interakcija između dva šetača započinje kada su unutar zadane udaljenosti (u simulaciji je korištena vrijednost od dvije tjelesne duljine mušice, koja iznosi približno 5 milimetara, normalizirana na $[0, 1]$ interval) i njeno trajanje se povećava za jedan za svaki uzastopni trenutak[4] promatranja u kojem ostaju unutar zadane vrijednosti. Jednom kada se šetači odmaknu za više od 5 milimetara, interakcija završava i podatci o njoj (mušice koje su interagirale, početni trenutak, trajanje, završni trenutak) se spremaju u podatkovni okvir.
Nakon što se izračunaju sve interakcije među svim parovima šetača, od istih se kreiraju mreže[5] koje za svaki par šetača bilježe broj međusobnih interakcija, kronološki poredanu listu trajanja svih interakcija i sveukupno trajanje interakcija.
Nad dobivenim mrežama računaju se mjere koje omogućavaju usporedbu simulirane i stvarne mreže.

[4] trenutak u ovom konteksu ovisi o stopi uzorkovanja, s obzirom da je bazična frekvencija simulacije 24 puta po sekundi, trajanje interakcije se mjeri u $\frac{1}{24}$ sekunde

[5] ili grafovi, u kontekstu diskretne matematike

4 Implementacija u programskom kodu

U nastavku su opisane implementacije dijelova stvarnog sustava. Cijela simulacija je odrađena u programskom jeziku Python koristeći objektno-orijentiranu paradigmu.
U nastavku su navedene napomene vezane uz metodu simuliranja:

- Sve numeričke vrijednosti u programu normalizirane su na interval $[0, 1]$.
- Vrijeme izvedbe simulacije indirektno je reprezentirano kao broj koraka mušica koji se simulira.
- Pozicija mušice u određenom koraku označena je kao točka u dvodimenzionalnom kartezijevom sustavu.
- Arena koja ograničava kretanje mušica implementirana je pomoću kružnice sa zadanim promjerom. Radi jednostavnosti, polumjer kružnice iznosi 0.5, a središte kružnice je zadano na $O(0.5, 0.5)$.
- Osnovni modeli koji se koriste u programu su model mušice, kartezijeve dvodimenzionalne točke i korak pomaka mušice.

Cjeloviti programski kod dostupan je na otvorenom repozitoriju na poveznici.

4.1 Model točke

Model točke implementiran je klasom `Point`.
Klasa `Point` sastoji se od jednog konstruktora koji kao parametre uzima koordinate apscise i ordinate i pridodaje ih varijablama instance i preopterećenim operatorima zbrajanja i različitosti.

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return self.x + other.x, self.y + other.y

    def __ne__(self, other):
        return self.x != other.x or self.y != other.y
```

Izvorni kod 1: Klasa `Point`

4.2 Model koraka

Model koraka implementiran je klasom `Step`, a modelira razmak i smjer između dvaju točaka u stvarnom sustavu. Smjer je definiran kao mjera kuta između dužine definirane dvaju točkama i apscise u pozitivnom smjeru. Poprima vrijednosti iz intervala $[0, 1]$ i iskazan je u **radijanima**.

```
class Step:
    def __init__(self, direction, distance):
        self.direction = direction
        self.distance = distance
```

Izvorni kod 2: Klasa `Step`

4.3 Model mušice

Model mušice implementiran je klasom `Fly`. Klasa `Fly` sadrži sljedeće atribute:

- Atribut `id_iterator` koji je **klasni objekt** [15] koji svakoj mušici pridružuje jednoznačni identifikator.
- Atribut `id` - jedinstveni identifikator.
- Atribut `point_list` koji sadrži sve dosadašnje pozicije mušice.
- Atribut `current_point` koji sprema informacije o trenutnoj poziciji mušice.

Klasa `Fly` ima jedan konstruktor s obveznim parametrom `point` koji predstavlja početnu točku. Tijelo konstruktora postavlja vrijednost trenutne točke na proslijeđenu, dodaje ju u listu svih točaka i generira identifikator pomoću gore definiranog iteratora.

```
import itertools

class Fly:
    id_iterator = itertools.count(start=1)

    def __init__(self, point):
        self.current_point = point
        self.point_list = [point]
        self.id = next(self.id_iterator)
```

Izvorni kod 3: Konstruktor klase `Fly`

Metoda `get_target_point` kao parametar prima objekt tipa `Step` s kojim se, uz pomoć trigonometrijskih funkcija sinusa i kosinusa, dobijaju pomaci po apscisi i ordinati. Povratna vrijednost metode je točka translatirana za izračunate pomake.

```
def get_target_point(self, step):
    target_x = step.distance * math.cos(step.direction)
    target_y = step.distance * math.sin(step.direction)

    return Point(self.current_point.x + target_x,
    self.current_point.y + target_y)
```

Izvorni kod 4: Metoda `get_target_point`

Metoda `move_to` kao ulazni parametar prima ciljnu točku mušice i pomiče mušicu na novu točku, pritom ju dodajući u listu svih posjećenih točaka.

```
def move_to(self, target_point):
    self.current_point = target_point
    self.point_list.append(self.current_point)
```

Izvorni kod 5: Metoda `move_to`

Metoda `move_in_sequence` kao ulazne parametre prima listu koraka. Metoda za svaki korak u listi generira odredišnu točku, provjerava je li točka u krugu i, ako nije, generira novu točku dok ne dobije točku koja jest. Nakon toga poziva metodu `move_to` s odredišnom točkom kao parametrom.

```
def move_in_sequence(self, step_list):
    calc_helper = CalculationHelper()
    data_generator = DataGenerator()

    for step in step_list:
        target_point = self.get_target_point(step)
        while target_point == self.current_point or not
        calc_helper.is_point_in_circle(target_point):
            tempDirection = calc_helper.normalize_angle(step.direction)
            tempDirection = tempDirection + 0.25
            if (tempDirection > 1):
                tempDirection = tempDirection - 1
            step.direction = calc_helper.angle_from_normalized(tempDirection)
            target_point = self.get_target_point(step)
        self.move_to(target_point)
```

Izvorni kod 6: Metoda `move_in_sequence`

4.4 Pomoćne klase (engl. *Helperi*)

U objektno-orijentiranoj paradigmi princip jedne odgovornosti nužna je smjernica za održivost čitkog koda i modularnosti [26].
Pomoćne klase su klase koje se koriste u svrhu grupiranja često korištenih operacija ili moduliranja funkcionalnosti koje ne podliježu ostalim definiranim modelima unutar aplikacije.

U nastavku su opisane pomoćne klase korištene u simulaciji:

- `CalculationHelper` - pomaže s općenitim kalkulacijama kao što su euklidska udaljenost, normaliziranje koordinata u specifični interval, pretvaranje stupnjeva u radijane i slično.
- `SimulationConstantHelper` - sadrži globalne konstante koje pomažu pri izvođenju programa, kao što su broj mušica, skalirani i stvarni radijus arene i slilčno.
- `DirectoriesConstantHelper` - sadrži globalne konstante koje sadrže putanje do direktorija korištenih u izvedbi simulacije.
- `DataGeneratorHelper` - generira testne podatke za testiranje funkcionalnosti programa.
- `DataHelper` - odrađuje operacije nad podatkovnim okvirima *python* biblioteke **pandas** i operacije nad mrežama (grafovima) biblioteke **networkx**.
- `PlotHelper` - komunicira sa `matplotlib.pyplot` modulom; iscrtava putanje kretanja mušice i kružnicu koja simulira arenu.
- `UtilityHelper` - odrađuje generične operacije nad direktorijima, kao što su dobavljanje direktorija za spremanje podataka, brisanje podataka iz istog, dobavljanje trenutnog vremena koje se koristi za imenovanje datoteka i slično.

Valja primijetiti da su u programskom kodu klase imenovane po `CamelCase (CapWords)` konvenciji, dok su datoteke u kojima se nalaze imenovane `snake_case` konvencijom. Ovaj pristup dizajnu je namjeran i prati PEP 8 praksu imenovanja.

Pivotalna klasa u programu je klasa `SimulationHelper` koja povezuje sve ostale pomoćne klase i modele kako bi izvršila simulaciju. Konstruktor klase podatkovnim članovima klase pridodaje vrijednosti koje se nalaze u pomoćnoj klasi `ConstantHelper`.

```
class SimulationHelper:
    def __init__(self):
        calc_helper = CalculationHelper()

        self.step_number = STEP_NUMBER
```

```
        self.arena_radius = ARENA_RADIUS_SCALED
        self.fly_list = [Fly(calc_helper.generate_point_in_circle()) for x
        in range(FLY_NUMBER)]
        self.distance_threshold = INTERACTION_DISTANCE_THRESHOLD
```

Izvorni kod 7: Konstruktor klase `SimulationHelper`

Valja primijetiti da se lista mušica popunjava **list comprehension** [18] metodom u kojoj se broj mušica proslijeđuje parametrom i za svaku se početna pozicija postavi na nasumičnu točku unutar arene koja je generirana metodom `generate_point_in_circle` iz pomoćne klase `CalculationHelper`.

Klasa `SimulationHelper` sadrži ključne članove za generiranje šetnji, spremanja podataka i vizualizaciju:

Metoda `generate_walks` za svaku mušicu u listi mušica generira listu nasumičnih koraka kroz koje provodi mušicu.

```
    def generate_walks(self):
        print("Generating walks...")
        dataGenerator = DataGenerator()

        for fly in self.fly_list:
            dataGenerator.generate_random_steps(self.step_number)
            fly.move_in_sequence(dataGenerator.steps)

        print("Walks generated.")
```

Izvorni kod 8: Metoda `generate_walks`

Metoda `export_fly` kreira podatkovni okvir, popunjava ga informacijama mušice koja je proslijeđena kao parametar i zatim ga zapisuje u odredišni direktorij za podatke. Ime datoteke je formatirano na sljedeći način:
`flyID_YYYY-MM-DD--HH-MM-SS`
Osim zapisivanja podatkovnih okvira, metoda provjerava postoje li direktoriji za spremanje animacija, statičnih vizualizacija šetnji i podataka o šetnjama mušica i, ako ne postoje, kreira ih. Nakon toga sprema podatke o šetnjama i poziva metodu `export_plot` za vizualizacije šetnji.

```
    def export_fly(self, fly):
        plot_helper = PlotHelper()
        x_coordinates = [point.x for point in fly.point_list]
        y_coordinates = [point.y for point in fly.point_list]
```

```
        plot_helper.set_coordinates(x_coordinates, y_coordinates)

        dict = {"id" : fly.id, "pos x": x_coordinates, "pos y": y_coordinates}
        df = pd.DataFrame(dict)
        os.makedirs(MOVEMENT_DIR, exist_ok=True)
        os.makedirs(ANIMATION_DIR, exist_ok=True)
        os.makedirs(PLOT_DIR, exist_ok=True)

        movement_filename = MOVEMENT_DIR + "/" + str(fly.id) + "_" +
        get_current_time() + ".csv"
        df.to_csv(movement_filename)

        plot_filename = PLOT_DIR + "/" + str(fly.id) + "_" +
        get_current_time() + ".png"
        plot_helper.export_plot(plot_filename)
```

Izvorni kod 9: Metoda `export_fly`

Metoda `export_all` poziva metodu `export_fly` nad svim mušicama u simulaciji i čisti prethodno generirane podatke.

```
    def export_all(self):
        clear_all()

        for fly in self.fly_list:
            self.export_fly(fly)
```

Izvorni kod 10: Metoda `export_all`

Metoda `export_all_fly_interactions` kreira *python* riječnik koji kao ključ uzima identifikator mušice, a kao vrijednosti kreira podatkovni okvir sastavljen od koordinata apscise i ordinate. Nad tim riječnikom se pozivaju metode iz pomoćne klase `DataHelper` koje generiraju udaljenosti među mušicama i njihove interakcije i spemaju ih u .csv formatu u za to predviđen direktorij. Od generiranih udaljenosti se kreiraju i spremaju mreže.

```
    def export_all_fly_interactions(self):
        fly_dict = {fly.id: pd.DataFrame(
        {"pos x":
            [point.x for point in fly.point_list],
         "pos y":
            [point.y for point in fly.point_list]})
```

```
            for fly in self.fly_list}

        all_flies_distances = distances_between_all_flies(fly_dict)
        all_flies_interactions = get_fly_interactions(all_flies_distances,
        self.distance_threshold)

        os.makedirs(OUTPUT_DIR, exist_ok=True)

        all_flies_distances.to_csv(OUTPUT_DIR + "/distances.csv", index =
        True)
        all_flies_interactions.to_csv(OUTPUT_DIR + "/interactions.csv",
        index = False)
        save_as_graph(all_flies_interactions)
```

Izvorni kod 11: Metoda `export_all_fly_interactions`

4.5 Funkcija `main`

Funkcija `main` programa je, u duhu objektno-orijentirane paradigme, vrlo minimalistička: Inicijalizira se objekt klase `SimulationHelper` nakon čega se šetnje generiraju i spremaju. Na osnovu generiranih šetnji se računaju i spremaju udaljenosti mušica i njihove interakcije na osnovu kojih se generiraju i spremaju mreže. Poziva se metoda `animate_simulation` koja animira simulaciju ako je vrijednost neke od globalnih pomoćnih varijabli za animiranje postavljena na `True`. U ulaznoj točki programa simulacija se pokreće željeni broj puta, računaju se i vizualiziraju mjere mreža.

```
def main():
    simulation = SimulationHelper()
    simulation.generate_walks()
    simulation.export_all()
    simulation.export_all_fly_interactions()
if __name__=="__main__":
    plot_helper = PlotHelper()

    for i in range(1000):
         main()

    export_all_graphs_global_measures()
    plot_helper.plot_measures()
```

Izvorni kod 12: Funkcija `main`

5 Rezultati

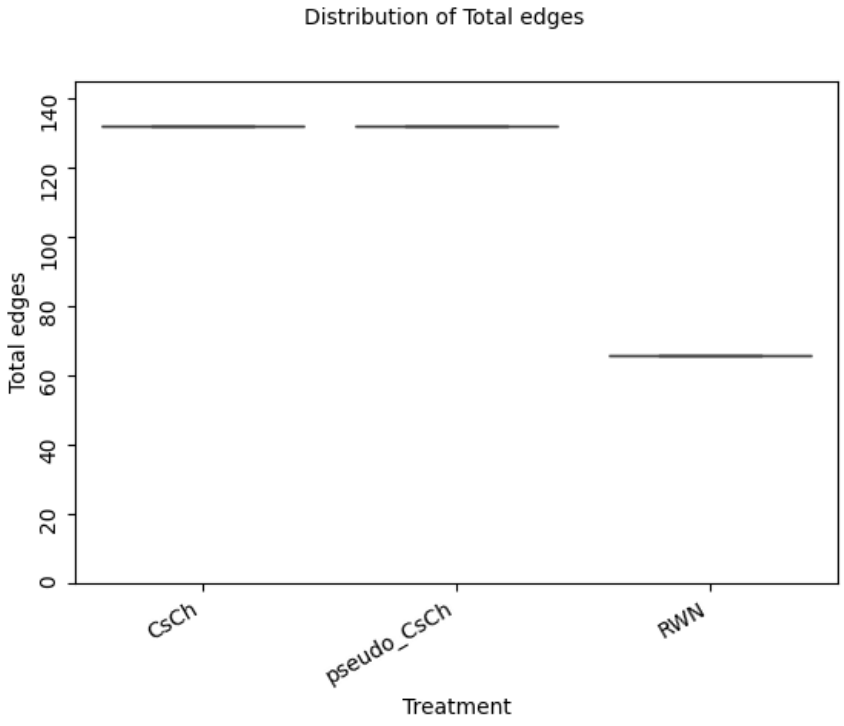
U nastavku su opisane razlike između simuliranih podataka s dva skupa stvarnih podataka promatranja koje ćemo nadalje zvati **tretmani**:

- Tretman *CsCh* - promatranje dvanaest običnih mušica unutar jedne arene. Ovo je osnovni (engl. *baseline*) tretman.
- Tretman *pseudo_CsCh* - kreira se na način da se iz *CsCh* tretmana nasumično odabire 12 promatranja, a zatim se iz svakog od tih promatranja nasumično odabire jedna mušica. Na ovaj način se od 12 nepovezanih mušica kreira novo promatranje. S obzirom da se promatraju mušice iz odvojenih promatranja, ovaj tretman izostavlja socijalni faktor koji je potencijalno utjecao na kretanje mušica.

Simulirane podatke ćemo nadalje zvati *RWN* (engl. *random walking agents*) tretman.

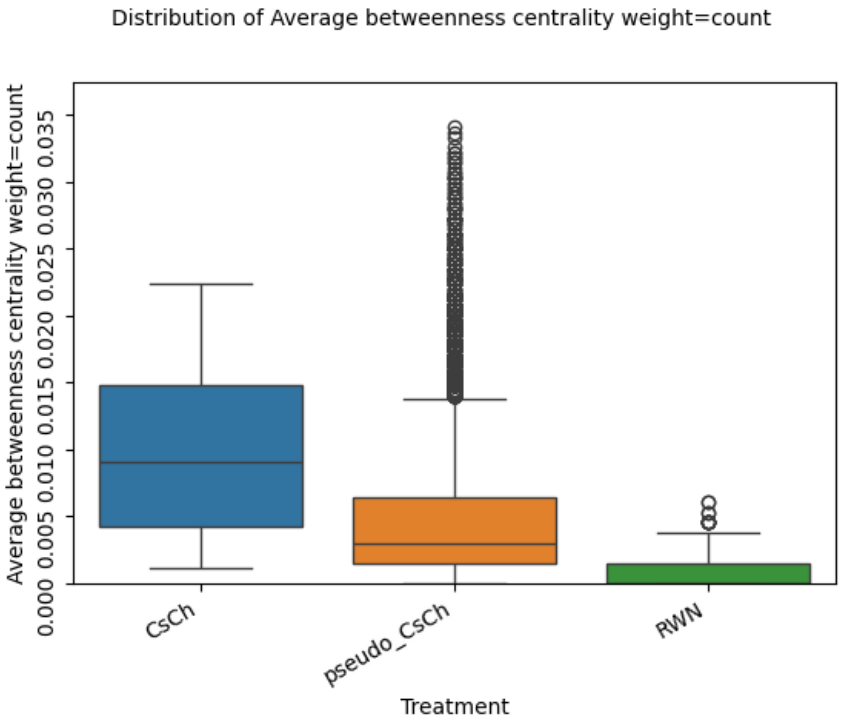
Sva promatranja su trajala 20 minuta, sa uzorkovanjem koraka od 24 puta u sekundi. Podatci se uspoređuju na uzorku od 1000 promatranja i na osnovu mjera mreže.

U tretmanima *CsCh* i *pseudo_CsCh* korišteni su usmjereni grafovi, dok je kod tretmana *RWN* korišten neusmjeren graf. Na Slici 1. prikazan je **ukupan broj bridova (interakcija)** među mušicama u stvarnim podatcima koji iznosi 132 ($12 \cdot 11$, svaka mušica se povezuje sa svim ostalima), dok je kod tretmana *RWN* broj bridova jednak 66 ($\frac{12 \cdot 11}{2}$, svaka mušica se povezuje sa svim ostalima, ali se duple interakcije među istim parom mušica računaju kao ista interakcija). Iz ovih podatak nužno slijedi da je mreža **potpuno povezana**.



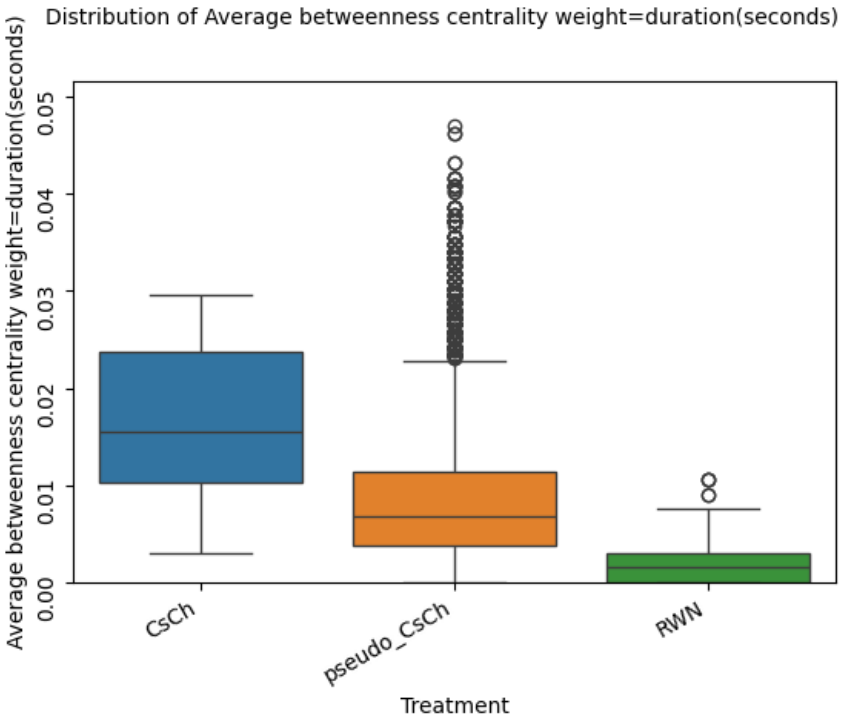
Slika 1: Distribucija interakcija mušica

Distribucija prosječne centralnosti posredovanja po broju interakcija izračunata je na način da se za svaki čvor unutar jedne mreže izračuna centralnost posredovanja i uzme se njihov prosjek. Na Slici 2. vidljiva je distribucija tih prosjeka za svaku promatranu mrežu. Distribucija se kreće u intervalu $[0, 0.035]$, a vrijednosti su najgušće rapoređene u intervalu $[0, 0.015]$. Prikazane vrijednosti nad *CsCh* tretmanom pokazuje da većina mušica interagira s ispodprosječnim brojem zajednica mušica, dok kod *pseudo_CsCh* tretmana rubni postotci mušica imaju interakciju s većinom zajednica mušica ili vrlo malo njih, a većina mušica interagira s manje zajednica nego u *CsCh* tretmanu. Kod tretmana *RWN* jako mali broj mušica interagira s ostalim zajednicama mušica, što je moguća posljedica toga da postoji manje zajednica mušica, odnosno da su mušice generalno zbijenije (pogledati distribuciju centralnosti blizine).



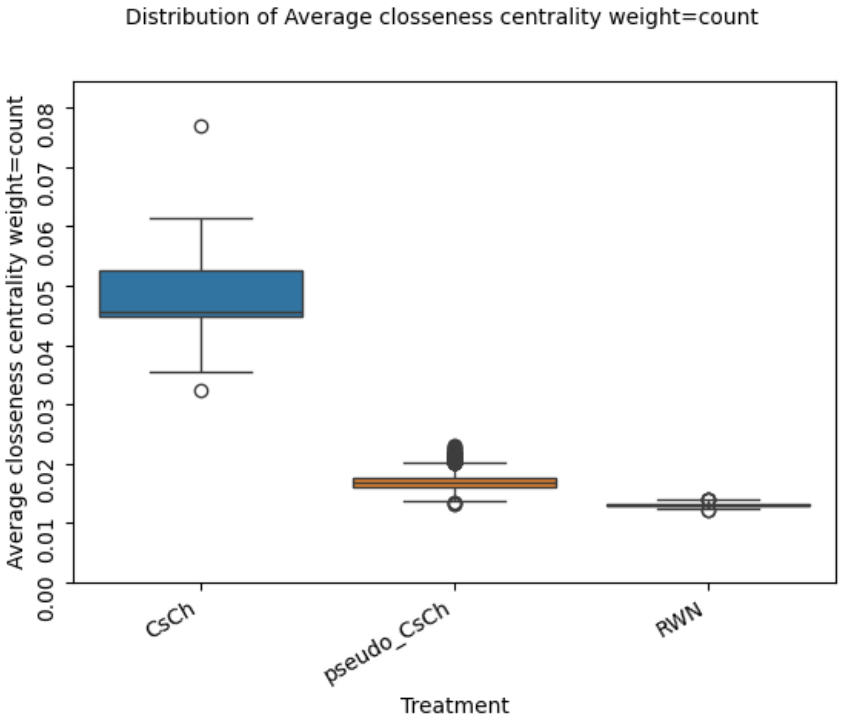
Slika 2: Distribucija prosječne centralnosti posredovanja po broju interakcija

Distribucija prosječne centralnosti posrednosti po duljini interakcija (vidljiva na Slici 3.) se minimalno razlikuje od iste distribucije po broju interakcija i potvrđuje prethodno izvedene zaključke. Vrijednosti distribucije su nešto drukčije raspoređene i kreću se u intervalu $[0, 0.05]$ dok su vrijednosti najgušće zbijenije u intervalu $[0, 0.025]$.



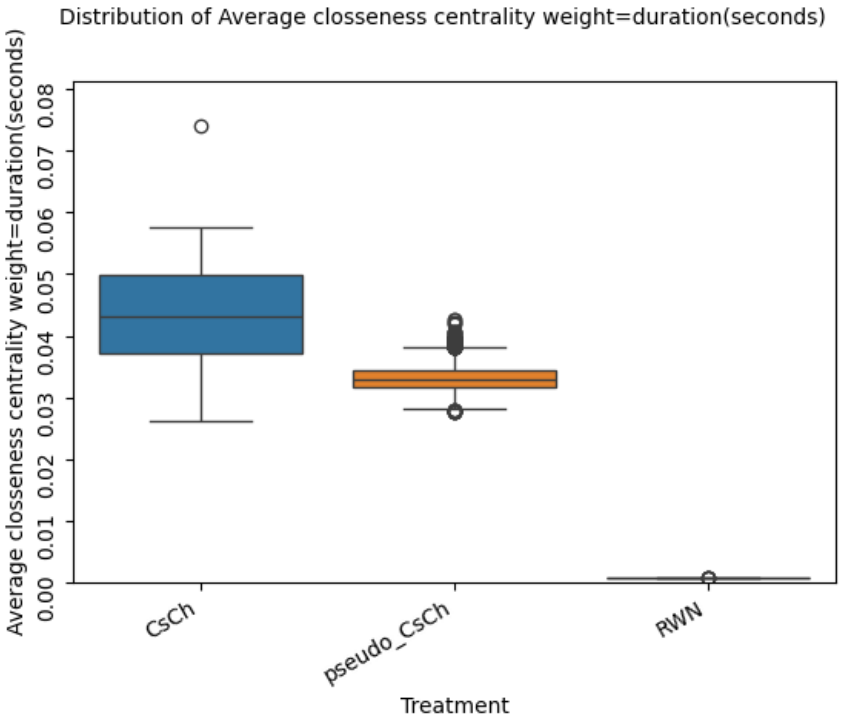
Slika 3: Distribucija prosječne centralnosti posredovanja po duljini interakcija

Distribucija prosječne centralnosti blizine po broju interakcija prikazana je na Slici 4. i nad *CsCh* tretmanom pokazuje da su mušice većinom raštrkane, a prosječan broj interakcija između mušica je ispodprosječan. Vrijednosti ove mjere se za tretmane *pseudo_CsCh* i *RWN* nalaze u vrlo uskom intervalu $[0.01, 0.021]$ dok se vrijednosti tretmana *CsCh* nalaze u intervalu $[0.036, 0.062]$. Vrijednosti mjere nad tretmanom *RWN* i *pseudo_CsCh* tretmanom su međusobno slične i pokazuju da su mušice puno bliže jedne drugima, odnosno međusobno imaju puno više interakcija nego što je to u *CsCh* tretmanu.



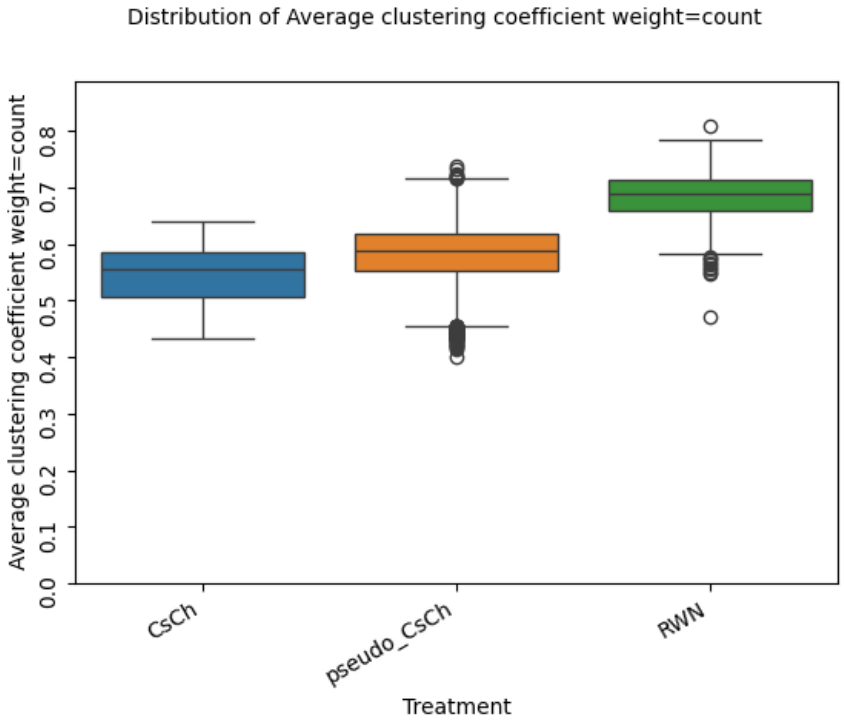
Slika 4: Distribucija prosječne centralnosti blizine po broju interakcija

Distribucija prosječne centralnosti blizine po duljini interakcija (prikazana na Slici 5.) s druge strane pokazuje interesantnu promjenu: centralnosti kod *CsCh* i *pseudo_CsCh* tretmana su međusobno sličnije i bliže su aritmetičkoj sredini distribucije, odnosno trajanje interakcija između svih mušica je relativno izjednačeno, dok vrijednosti mjere nad tretmanom *RWN* pokazuju da trajanje interakcija među mušicama traje puno dulje. Vrijednosti distribucije za tretman *RWN* se kreću oko nule, a vrijednosti distribucije za tretmane *CsCh* i *pseudo_CsCh* kreću se u intervalu $[0.028, 0.058]$ i najgušće su raspoređene u intervalu $[0.035, 0.05]$.



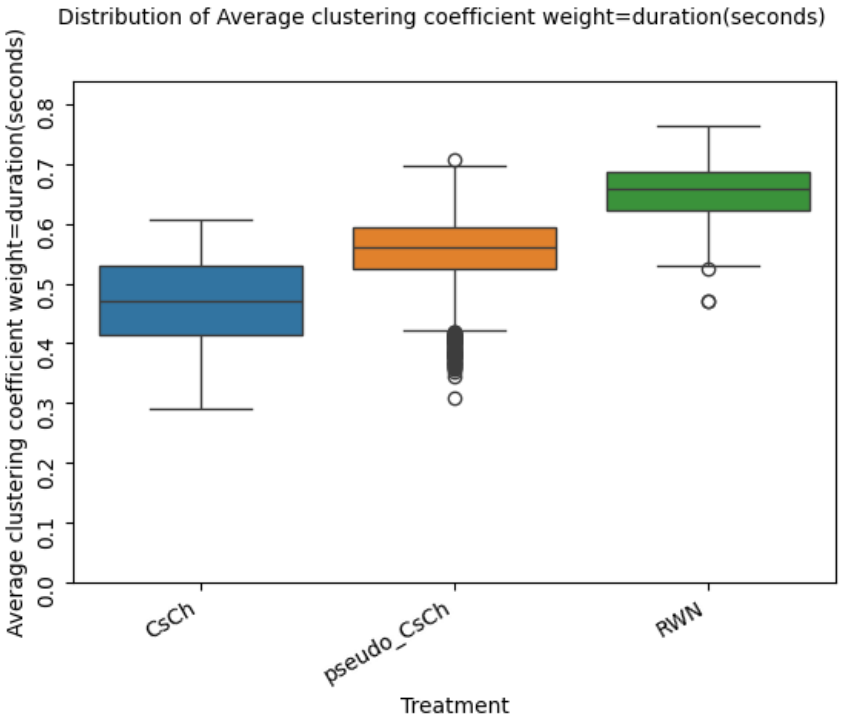
Slika 5: Distribucija prosječne centralnosti blizine po duljini interakcija

Distribucija prosječnog koeficijenta grupiranja po broju interakcija prikazana je na Slici 6. i pokazuje mala odskakanja među tretmanima *CsCh* i *pseudo_CsCh*. Vrijednosti distribucije nad tretmanima *CsCh* i *pseudo_CsCh* nalaze se u intervalu $[0.42, 0.71]$ i teže ka 0.6 od maksimalnih 0.8, što se tumači kao odlična povezanost mušica u klike. Vrijednosti mjere nad tretmanom *RWN* nalaze se u intervalu $[0.59, 0.8]$ i teže ka 0.7 od maksimalnih 0.8, što se tumači kao skoro potpuno povezivanje mušica u klike.



Slika 6: Distribucija prosječnog koeficijenta grupiranja po broju interakcija

Distribucija prosječnog koeficijenta grupiranja po trajanju interakcija prikazana je na Slici 7. i pokazuje male razlike od grupiranju po broju interakcija. Vrijednosti distribucije nad tretmanima *CsCh* i *pseudo_CsCh* kreću se u intervalu $[0.3, 0.7]$ i najgušće su raspoređene u intervalu $[0.41, 0.6]$ dok se vrijednosti nad *RWN* tretmanom nalaze u intervalu $[0.54, 0.77]$ i najgušće su raspoređene u intervalu $[0.65, 0.7]$. Vrijednosti mjere nad tretmanima *CsCh*, *pseudo_CsCh* i *RWN* sada teže prema 0.47, 0.58 i 0.68 od maksimalnih 0.8, respektivno, što se tumači na način da su prosječne duljine interakcija u mreži kraće u usporedbi s brojem interakcija [29].



Slika 7: Distribucija prosječnog koeficijenta grupiranja po trajanju interakcija

6 Zaključak

Izrada modela i simuliranje kompleksnih sustava jedna je od vodećih metoda u znanstvenom istraživanju u brojnim disciplinama, koristeći empirijske podatke, heuristiku i druge tehnike za uštedu resursa i vremena koji bi se alternativno trebali utrošiti za postizanje sličnih rezultata. Simuliranjem kretanja vinskih mušica u dvodimenzionalnom prostoru pomoću nasumičnih šetača došli smo do podataka za koje bismo inače trebali uložiti vremenske i fizičke resurse za postavljanje, održavanje i promatranje ponašanja mušica u izoliranim uvjetima. Kreiranjem mreža od dobivenih podataka i uspoređivanjem istih na osnovu mjera nad mrežama pokazano je da su dobiveni podatci mjerodavni i približno prikazuju realno stanje sustava. Glavne razlike dobivene usporedbom su direktna posljedica nepotpune povezanosti nastalih mreža, što bi se u sljedećim iteracijama istraživanja moglo nadograditi implementacijom tendencije mušica da se kreću prema drugim jedinkama ovisno o socijalnim značajkama (ekstrovertiranosti, introvertiranosti i slično).

7 Literatura

[1] D. B. Roberts, ”*Drosophila melanogaster*: the model organism.” wiley.com. [Online] Dostupno na: https://onlinelibrary.wiley.com/doi/full/10.1111/j.1570-8703.2006.00474.x (pristupljeno: 26.6.2024.)

[2] Z. Mirzoyan i drugi, ”*Drosophila melanogaster*: A Model Organism to Study Cancer,” in *Model Organisms: A Precious Resource for Understanding of the Molecular Mechanisms Underlying Human Physiology and Disease*, vol. 10, M. G. Giansanti, 2019. Dostupno na: https://www.frontiersin.org/journals/genetics/articles/10.3389/fgene.2019.00051/full

[3] N. Alwash, A. M. Allen, M. B. Sokolowski, and J. D. Levine, ”The *Drosophila melanogaster* foraging gene affects social networks,” in *Journal of Neurogenetics*, 2021, pp. 249-261. Dostupno na: https://pubmed.ncbi.nlm.nih.gov/34121597/

[4] G. Liu i drugi, ”A simple computer vision pipeline reveals the effects of isolation on social interaction dynamics in Drosophila.” A. A. Faisal, Imperial College London, 2018. Dostupno na: https://doi.org/10.1371/journal.pcbi.1006410

[5] F. Rigo, A. Filošević Vujnović, M. Petrović, K. Jovic, and R. Andretić Waldowski, ”Locomotor sensitization modulates voluntary self-administration of methamphetamine in Drosophila melanogaster” in *Addiction Biology*, 2020, doi: 10.1111/adb.12963. Dostupno na poveznici.

[6] M. Petrović, A. Meštrović, R. Andretić Waldowski, and A. Filošević Vujnović, ”A network-based analysis detects cocaine-induced changes in social interactions in Drosophila melanogaster,” in *PLOS ONE* , 2023, doi: 10.1371/journal.pone.0275795. Dostupno na poveznici.

[7] Y. Bar-Yam, *General Features of Complex Systems.* Encyclopedia of Life Support Systems, EOLSS UNESCO Publishers, Oxford, UK, 2002.

[8] J. Polkinghorne, ”Reductionism,” in *Interdisciplinary Encyclopedia of Religion and Science*, G. Tanzella-Nitti, I. Colagé, and A. Strumia, 2002, doi: 10.17421/2037-2329-2002-JP-2. Dostupno na: https://inters.org/reductionism

[9] G. Wallentin, ”Spatial Simulation,” UNIGIS distance learning program in Geoinformatics, University of Salzburg, Salzburg, 2024. [Online]. Dostupno na: https://unigis-salzburg.github.io/Opt_Spatial-Simulation/ (pristupljeno: 15.7.2024.)

[10] X. Feng, L. Jiaying, N. Hansong, F. Yonghao, W. Liangtian, and K. Xiangjie, ”Random Walks: A Review of Algorithms and Applications.”, arxiv.org. [Online]

Dostupno na: https://arxiv.org/pdf/2008.03639 (pristupljeno: 10.6.2024.)

[11] L. Hardesty, ”Explained: Linear and nonlinear systems.” mit.edu. [Online]
Dostupno na: https://news.mit.edu/2010/explained-linear-0226 (pristupljeno: 16.6.2024.)

[12] T. O’Connor, ”Emergent Properties,” in *Stanford Encyclopedia of Philosophy*, Edward N. Zalta, 2012. [Online].
Dostupno na: https://plato.stanford.edu/archives/spr2012/entries/properties-emergent/ (pristupljeno: 8.8.2024.)

[13] A. Gupta and S. Anish, ”Insights from Complexity Theory: Understanding Organisations better.”, iimb.ac.in. [Online].
Dostupno na: https://tejas.iimb.ac.in/articles/12.php (pristupljeno: 10.8.2024.)

[14] R. Červenka, I. Trenčanskỳ, M. Calisti, and D. Greenwood, “AML: Agent Modeling Language Toward Industry-Grade Agent-Based Modeling.” 2005, pp. 31-46, doi: 10.1007/978-3-540-30578-1_3.

[15] ”Klasni objekti”, službena *Python* dokumentacija.
Dostupno na: https://docs.python.org/3/tutorial/classes.html (pristupljeno: 15.6.2024.)

[16] ”random walk.” britannica.com. [Online].
Dostupno na: https://www.britannica.com/science/random-walk (pristupljeno: 22.6.2024.)

[17] ”Rešetka.” ihjj.hr. [Online].
Dostupno na: http://struna.ihjj.hr/naziv/resetka/33100/ (pristupljeno: 21.6.2024.)

[18] G. Van Rossum and F. L. Drake Jr, *Python reference manual.* CWI, 1995.

[19] J. D. Hunter, ”Matplotlib: A 2D Graphics Environment,” in *Computing in Science and Engineering*, 2007, vol. 9, no.3, pp. 90-95, doi: 10.1109/MCSE.2007.55.

[20] M. Newman, *Networks: An Introduction.* Oxford University Press, 2010.

[21] K. Cham AND J. Johnson, “Complexity Theory: A Science of Cultural Systems?” M/C Journal. [Online].
Dostupno na: https://doi.org/doi.org/10.5204/mcj.2672. (pristupljeno: 20.7.2024.)

[22] D. H. Meadows and D. Wright, *Thinking in Systems: A Primer.* Vermont: Chelsea Green Publishing, 2008.

[23] K. Pearson, ”The problem of the random walk”, Nature, vol. 72, no. 1867, 1905, p. 342.

[24] F. Spitzer, *Principles of random walk.* Springer, vol. 34, 2013.

[25] S. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability.* Cambridge University Press, 2009.

[26] S. Oloruntoba and A. S. Walia, ”SOLID: The First 5 Principles of Object Oriented Design.” digitalocean.com. [Online].
Dostupno na: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design (pristupljeno 22.6.2024.)

[27] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, ”Fast unfolding of communities in large networks”, J. Stat. Mech, 2008.
Dostupno na: https://arxiv.org/abs/0803.0476v2

[28] G. Pineda-Villavicencio, J. Ugon, and D. Yost, ”The Excess Degree of a Polytope”, SIAM Journal on Discrete Mathematics, vol. 32, 2018.
Dostupno na: https://epubs.siam.org/doi/10.1137/17M1131994

[29] ”clustering”, službena dokumentacija modula *NetworkX*.
Dostupno na: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.clustering.html (pristupljeno: 4.8.2024.)

Popis slika

Popis priloga