

TALLINNA TEHNIKAÜLIKOOL
Küberneetika Instituut

Taivo Lints
990849IAQDD

Flockheadz

Essee õppeaines "Modelleerimise alused"

Õppejõud: Jüri Engelbrecht

Tallinn 2006

Sissejuhatus

Tehisintellekti valdkonnas proovitakse luua rohkemal või vähemal määral (praeguse seisuga pigem teine variant) intellekti üles näitavaid tehissüsteeme. Mõnedel juhtudel proovitakse otseselt modelleerida bioloogilist aju (nt. tehisnärvivõrgud) või abstraktsemaid mõtlemisprotsesse (nt. sümbolloogikal põhinevad süsteemid), teinekord aga otsitakse inspiratsiooni kaugemalt. Näiteks võetakse mudeli aluseks loomakarjade, linnu-parvede või putukakolooniate käitumine. Käesolev essee kirjeldabki kõigepealt olemasolevaid sülemipõhiseid lähenemisi tehisintellekti valdkonnas ning seejärel pakub välja ühe seni veel kasutamata idee.

Modelleerimise seisukohast vaadates ei ole antud juhul tegemist klassikalise loodus-teadusliku lähenemisega, kus tavaliselt on eesmärgiks luua reaalse süsteemi käitumist ennustav mudel, vaid pigem suhteliselt vaba ja loova lähenemisega üldisema probleemi – mõnes konkreetses situatsioonis natukenegi "arukalt" käituva süsteemi loomine – lahendamisele. Et väljapakutud idee olemust ja tekkepõhjusti paremini mõista, tundus olevat vajalik pühendada suurem osa käesolevast tööst idee inspiratsiooniks olnud varasemate tööde kirjeldamisele.

Tehisintellekt

Tehisintellektiks (TI) nimetatakse tehisobjekti (tavaliselt arvuti) poolt üles näidatud intelligentsust [1], kusjuures intelligentsus ise tähendab võimet probleeme lahendada, arutleda, planeerida, abstraktselt mõelda, keelest ja ideedest aru saada, õppida [2]. Üldjoontes võib tehisintellekti uurijad jagada kaheks koolkonnaks: 1) tavapärase / konventsionaalne / sümbolipõhine / loogikapõhine / puhas tehisintellekt, ja 2) arvutuslik / numbripõhine tehisintellekt [1, 3].

Konventsionaalne TI kasutab (peaaegu) eranditult formaalseid meetodeid nagu loogika või puhas rakenduslik statistika. Jälgitakse, et arutlusmehhanismid oleksid tõestatavalt korrektsed ja et välja töötatud masinõppimise algoritmide lahenduseni jõudmise aeg oleks ettemääratav. Konventsionaalse tehisintellekti saavutuste / meetodite hulka kuuluvad näiteks:

- Ekspertsüsteemid – analüüsivad etteantud reeglibaasi alusel süsteemile antavat infot ja pakuvad välja diagnoosi või lahenduse.
- Juhtumipõhine arutlus (*case-based reasoning*) – proovib leida lahendusi uuele probleemile varasemate teadaolevate sarnaste probleemide lahendusi analüüsi-des.
- Bayes'i (uskumuste) võrgud – info on esitatud suunatud atsüklilise graafina, kus võrgusõlmedeks on muutujad ja sõlmi ühendavad kaared näitavad muutujate vahelisi sõltuvusi. Probleemide lahendamine toimub seda võrku analüüsisides.
- Käitumispõhine TI (*behavior based AI*) – proovib intelligentsi dekomponeerida alammoduliteks / kihtideks. Näiteks "alumised" kihid tegelevad konkreetsete alamprobleemidega ja annavad tulemused edasi "ülemistele" kihtidele, kus tehakse üldistused ja lõplikud otsused.

Arvutuslik TI põhineb enamasti iteratiivsel arenemisel / õppimisel ja info on tavaliselt esitatud numbrilisel kujul. Koondumise ja korrektsuse tõestatavusest olulisemaks peetakse praktikas töötava süsteemi loomist: ei lasta end liigselt häirida põhjalike teooriate puudumisest ja "häkitakse" töötav süsteem valmis eeldusega, et küll teooria kunagi hiljem järele jõuab. Arvutusliku tehisintellekti suundade hulka kuuluvad näiteks:

- Tehisnärvivõrgud – bioloogilistest närvivõrkudest inspireeritud tehissüsteemid.
- Hägasloogika süsteemid – arutlussüsteemid, kus objektide / väärtuste kuuluvus hulkadesse ei ole määratud tasemel "kuulub" või "ei kuulu", vaid kui palju kuulub (kuuluvuse väärtus asub lõigul 0..1; seejuures tuleb tähele panna, et tegu ei ole hulka kuulumise tõenäosusega).
- Evolutsioonilised algoritmid – bioloogilisest evolutsioonist inspireeritud optimeerimisalgoritmid.
- Sülemiintellekt (*swarm intelligence*) – suure hulga (tehis)indiviidide kollektiivse käitumise uurimine ja rakendamine.

Konventsionaalne ja arvutuslik TI ei ole loomulikult üksteisest järgalt eraldatud ja tihti proovitakse mõlema meetodeid omavahel kombineerida.

Sülemiintellekt

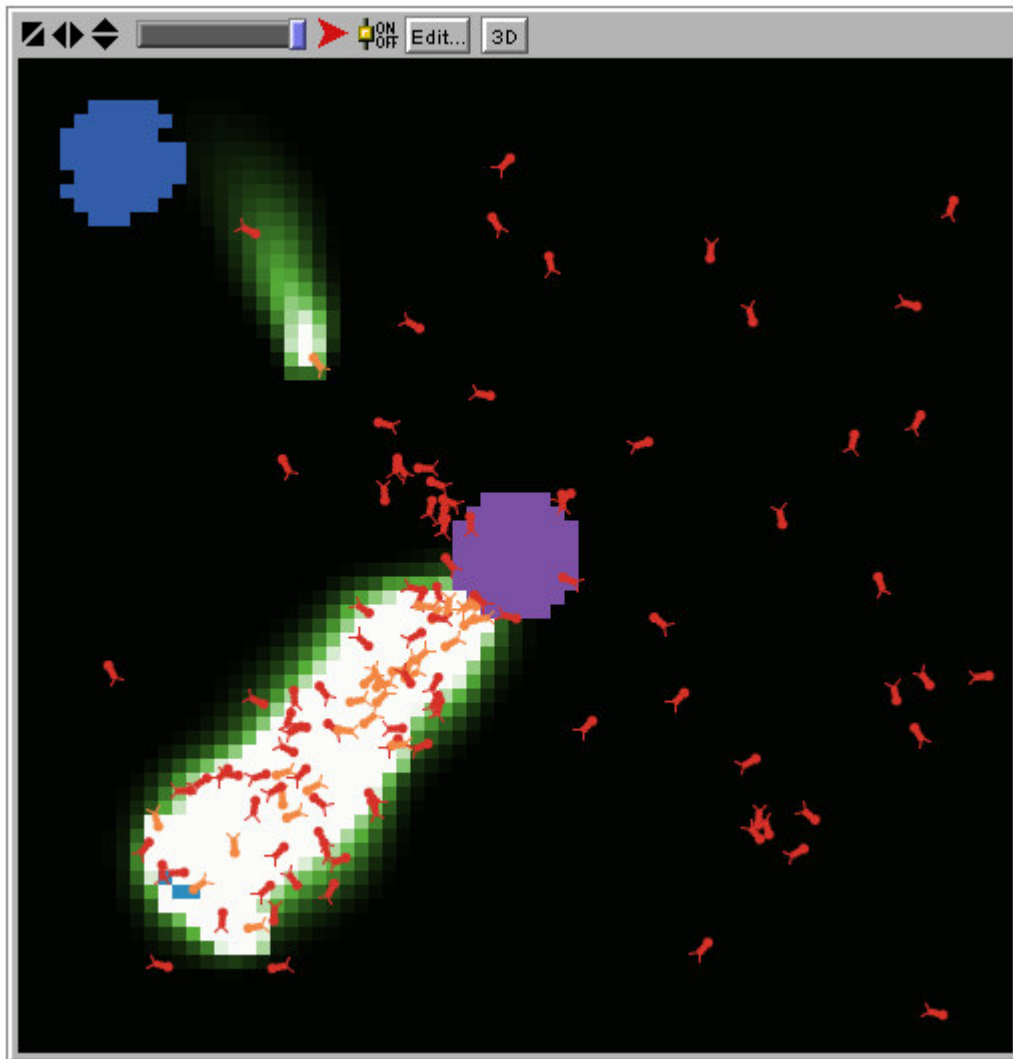
Sülemiintellekt (SI) on tehisintellekti üks harusid, mis on inspireeritud looduslike sülemite, karjade ja parvede käitumisest. Sülemiintellekti süsteemid koosnevad tavaliselt hulgast lihtsatest agentidest (s.t. suhteliselt autonoomsetest üksustest), mis on võimelised omavahel ja keskkonnaga suhtlema [4]. Looduslike sülemite vaatlemisel leitud põhimõtteid kasutatakse mitmesugustes erinevates rakendustes, näiteks robotikas, samuti arvutimängudes ja filmides parvede simuleerimisel. Sellistel juhtudel on tegu küllaltki otsese analoogiaga bioloogilise ja tehispärve vahel: eesmärgiks on panna tehisobjektid käituma suhteliselt ühtse kogumina: tehispärvena. Olemas on aga ka mõnevõrra abstraktsem lähenemisviis, kus tehissülemit kasutatakse konkreetsetele probleemidele lahenduse leidmiseks ilma, et sülemit ennast üldse hakatakski robotitena realiseerima ega isegi mitte ekraanil visualiseerima. Sellistest SI tehnikatest on hetkeseisuga ilmselt kõige rohkem rakendust leidnud sipelgakolooniaga optimeerimine (*Ant Colony Optimization, ACO*) ja osakeste parvega optimeerimine (*Particle Swarm Optimization, PSO*).

Sipelgakolooniaga optimeerimine

ACO on tõenäosuslik algoritm, mis on inspireeritud sipelgate käitumisest teede otsimisel pesast toiduni ja sobib selliste arvutuslike probleemide lahendamiseks, mis on taandatavad (graafil) heade teede leidmisele ühest punktist teise [5].

Toitu otsivate reaalsete sipelgate käitumine on üldjoontes järgmine. Otsimisega tegelevad sipelgad uitavad pesa ümbruses suhteliselt suvaliselt ringi. Kui sipelgas leiab toidu, siis võtab sellest kaasa niipalju, kui suudab, ja viib pessa. Seejuures eritab ta (maa)pinnale spetsiaalset feromooni (kemikaali). Kui mõni teine sipelgas (ja pesast tagasipöördudes ka esialgne sipelgas) leiab sellise feromooniraja, siis ta suure tõenäosusega lõpetab juhusliku ringirändamise ja asub rada järgima. Kui ta liigub õiges suunas ja jõuab toiduni, siis, analoogselt esialgse sipelgaga, võtab sellest osa, viib pessa ja samaaegselt lisab rajale feromooni. Selliselt võib hulga sipelgate korral välja kujuneda suure feromoonikontsentratsiooniga rada, mis omakorda meelitab ligi järjest uusi töösipelgaid, kes omakorda rada veelgi tugevdavad (joonis 1). Kuna sipelgate liikumine sisaldab isegi raja olemasolul väikest juhuslikku komponenti ja kuna rajaferomoonid aja jooksul aurustuvad, siis võib rada aegamööda esialgsest trajektooriga kõrvale triivida ning tavaliselt liigub ta just optimaalse (lühima) tee suunas.

ACO algoritmi korral kasutatakse reaalsete sipelgate sarnaselt käituvaid virtuaalseid objekte, mis liiguvad mööda probleemi kirjeldavat graafi või pinda ja otsivad head teed kahe punkti vahel, näiteks kahe serveri vahel arvutivõrgus või kahe vabriku / lao / kaupluse vahel logistilises võrgus.



Joonis 1. Simulatsioon toiduotsingul sipelgatest. Keskul asuva sipelgapesa ja vasakul all oleva toidu vahele on tekkinud tugeva feromoonikontsentratsiooniga sipelgarada. Pilt pärineb programmist NetLogo [6].

Osakeste parvega optimeerimine

PSO lähtub ideest, et kui karja üks liige juhtub nägema soodsat teed toidu vms.-ni, siis on tal võimalus teisi liikmeid sellest teavitada ja kõik isendid saavad kiiresti soodsasse piirkonda liikuda [7]. Selle modelleerimiseks (eesmärgiga kasutada tehissülemi parameetrite ruumist mingi eesmärgi jaoks hästi sobiva(te) punkti(de) leidmiseks) paigutatakse "otsinguruumi" algkiirusega osakesed, mis seejärel läbi ruumi lendavad. Osakesed suudavad meeles pidada parimat nähtud positsiooni ja sellest teisi sülemi liikmeid teavitada, samuti suudavad nad muuta oma kiirusvektorit vastavalt teadaolevatele headele positsioonidele. Suhtlemine osakeste vahel võib olla sõltuvalt mudelist globaalne, kus parim sülemi poolt seni leitud positsioon on teada kõigile, või lokaalne, kus iga osake saab suhelda ainult piiratud naabruskonnaga.

Valemid positsiooni ja kiiruse väärtuste muutmiseks igal ajasammul on järgnevad:

$$x \leftarrow x + v$$

$$v \leftarrow wv + c_1 r_1 (\hat{x} - x) + c_2 r_2 (\hat{x}_g - x)$$

w on konstant, mis iseloomustab osakese inertsi. Väärtus tavaliselt veidi väiksem ühest.

c_1 ja c_2 on konstandid, mis määravad, kui kiiresti osake heade positsioonide suunas pöörduv. Väärtused tavaliselt ühe ümbruses.

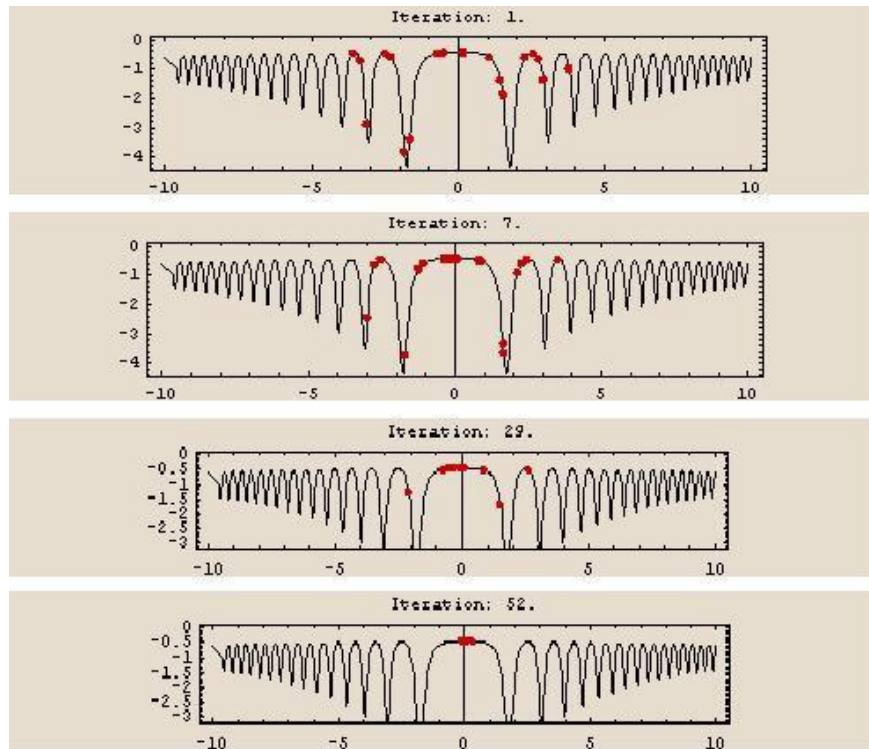
r_1 ja r_2 on juhuslikud väärtused lõigul $[0, 1]$.

\hat{x} on parim osakese poolt nähtud positsioon.

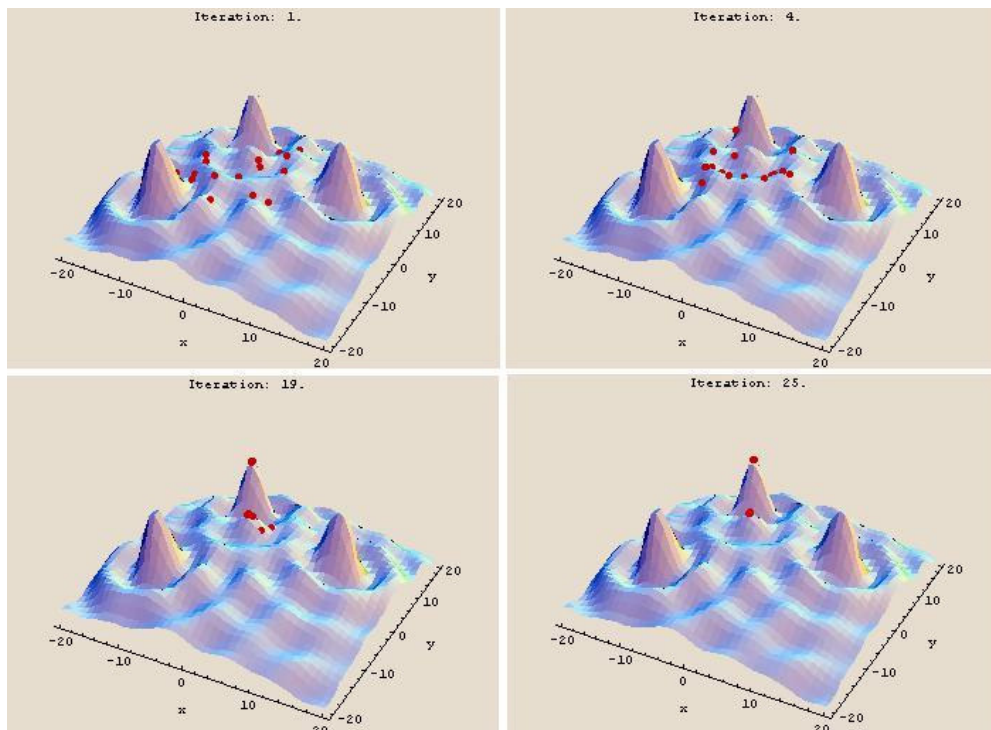
\hat{x}_g on globaalne parim sülemi poolt nähtud positsioon (lokaalse suhtluse korral on selle asemel lokaalne parim \hat{x}_l).

PSO algoritm:

- Anna iga osakese muutujatele x ja v juhuslik väärtus konkreetsele probleemile sobivast vahemikust.
- Algul igal osakesel $\hat{x} = x$
- \hat{x}_g algväärtuseks määra osakeste algpositsioonidest parim.
- Korda järgnevaid tegevusi seni, kuni positsiooni \hat{x}_g sobivus (*fitness*) ületab etteantud lävendi või iteratsioonide arv jõuab lubatud maksimumini.
 - Iga osakese jaoks:
 - ♦ Uuenda positsiooni x vastavalt eeltoodud valemile.
 - ♦ Arvuta uue positsiooni sobivus.
 - ♦ Kui see on parem kui \hat{x} sobivus, siis $\hat{x} = x$
 - ♦ Kui see on parem kui \hat{x}_g sobivus, siis $\hat{x}_g = x$
 - ♦ Uuenda kiirust v vastavalt eeltoodud valemile.



Joonis 2. Näide *PSO* töö kohta ühemõõtmelises ruumis. Graafiku x-koordinaat vastab ruumikoordinaadile ja y-koordinaat positsiooni sobivusele. Realse otsingu korral oleks muidugi teada ainult osakeste poolt läbitud positsioonide sobivusväärtused. [8]



Joonis 3. Näide *PSO* töö kohta kahemõõtmelises ruumis. Graafiku x- ja y-koordinaadid vastavad ruumikoordinaatidele ja z-koordinaat positsiooni sobivusele. Realse otsingu korral oleks jällegi teada ainult osakeste poolt läbitud positsioonide sobivusväärtused. [8]

Kui sobivusfunktsioon on ajas muutuv, siis ei piisa ühekordsest optimumi leidmisest, vaid otsinguruumil tuleb pidevalt "silma peal hoida". Lahendusena on pakutud näiteks osakeste jagunemist "liikideks", mis moodustavad alampopulatsioone kohalike optimumide ja nende ümbruste jälgimiseks [9].

Niisiis, praegu kasutusel olevad põhimeetodid sülemiintellekti selles harus, kus sülemil on "nähtamatu" probleemilahendaja roll, on:

- sipelgakolooniaga optimeerimine (*ACO*) – analoogia sipelgate toiduotsingutega, eesmärk leida optimaalseid teid ühest punktist teise;
- osakeste parvega optimeerimine (*PSO*) – analoogia maastikul liikuva otsimis-meeskonnaga, eesmärk leida parameetrite ruumist mingi eesmärgi suhtes optimaalseid punkte.

Kui *ACO* on üsna kitsalt "spetsialiseerunud" optimaalsete teede leidmisele, siis *PSO* on mõnevõrra laiem rakendusvaldkonnaga. Siiski on ka *PSO* kasutamisel mõningaid piiranguid. Näiteks on vaja iga sülemisse kuuluva osakese jaoks osata arvutada tema hetkepositsiooni sobivusväärtust. Üldjoontes võib öelda, et *PSO*, vähemalt praegu kasutusel olevatel kujudel, on rakendatav ikkagi üsna konkreetselt defineeritud ja määratud optimeerimisülesannete lahendamiseks.

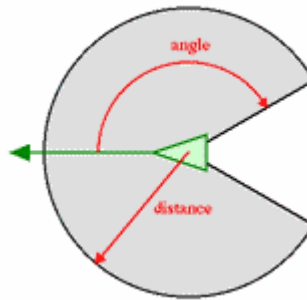
Järgnevas on välja pakutud üldisem sülemipõhine lähenemine mõningast intellekti olemasolu (loodetavasti) üles näitava süsteemi loomiseks, mis ei ole konstrueeritud väga konkreetset ülesannete klassi silmas pidades ega eeltoodud kahe meetodi asendamiseks, vaid pigem lootuses, et tulemus kusagil siiski rakendust leiab.

Flockheadz

Olles tutvunud kala- ja linnuparvede jms. simulatsioonidega ([10, 11, 12] jt.), tekkis mul mõte proovida luua süsteem, kus sülem on analoogselt *ACO* ja *PSO* 'ga "nähtamatu" probleemilahendaja rollis, kuid erinevalt neist kahest lähtuksid individid nimetatud parvesimulatsioonide aluseks olevatest reeglitest. Hetkeseisuga ei ole mulle teada ühtegi järgnevas kirjeldatuga analoogset süsteemi. Tõsi küll, ka minu poolt välja pakutu on alles idee tasemel – ei ole veel teostatud katseid idee mõistlikkuse ja rakendatavuse väljaselgitamiseks.

Elementaarset parvekäitumist on võimalik simuleerida väga lihtsa individipõhise mudeliga. Järgnevas on seda lühidalt kirjeldatud Craig Reynolds'i tööde põhjal [10].

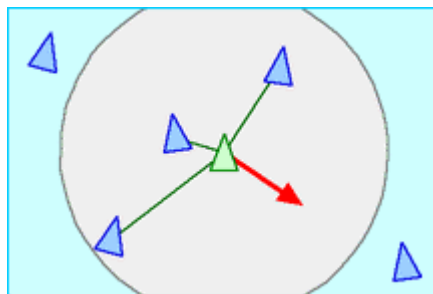
Individipõhise mudeli korral ei vaadelda parve kui ühtset tervikobjekti, vaid kui hulga (suhteliselt) autonoomsete üksikindividide käitumise tulemusena tekkivat nähtust (ka *ACO* ja *PSO* korral oli tegu individipõhise lähenemisega parve kujutamisele). Parve liikmeid nimetab Reynolds *boi*d'ideks (*bird-oid*, tuletatuna sõnast *bird* – lind). Iga boi di tegevust mõjutavad tema lähiümbruses olevad teised boi did, s.t. individid lähtuvad parvekäitumises lokaalsetest mõjuritest (see hoiab nõuded parveliidme tajumis- ja arvutusvõimetele suhteliselt väikesed ja võimaldab (looduses) väga suurte parvede teket, näiteks üle 17 miili pikkused migreeruvate heeringate parved [13]).



Joonis 4. Boid lähtub oma liikumise suunamisel piiratud lähiümbruses toimuvast (hall sektor). Selja taha jääb nn. "pime ala", kuhu boid lihtsalt ei "näe". [10]

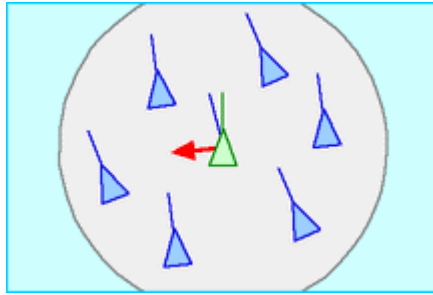
Elementaarse parvekäitumise tekkeks vajalikke reegleid on kolm:

1. Kokkupõrgete ja liigse läheduse vältimine:



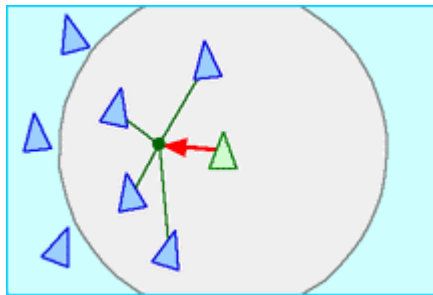
Joonis 5. Boid üritab vältida teistele liiga lähedale sattumist. [10]

2. Joondumine naabrite keskmise liikumissuuna järgi:



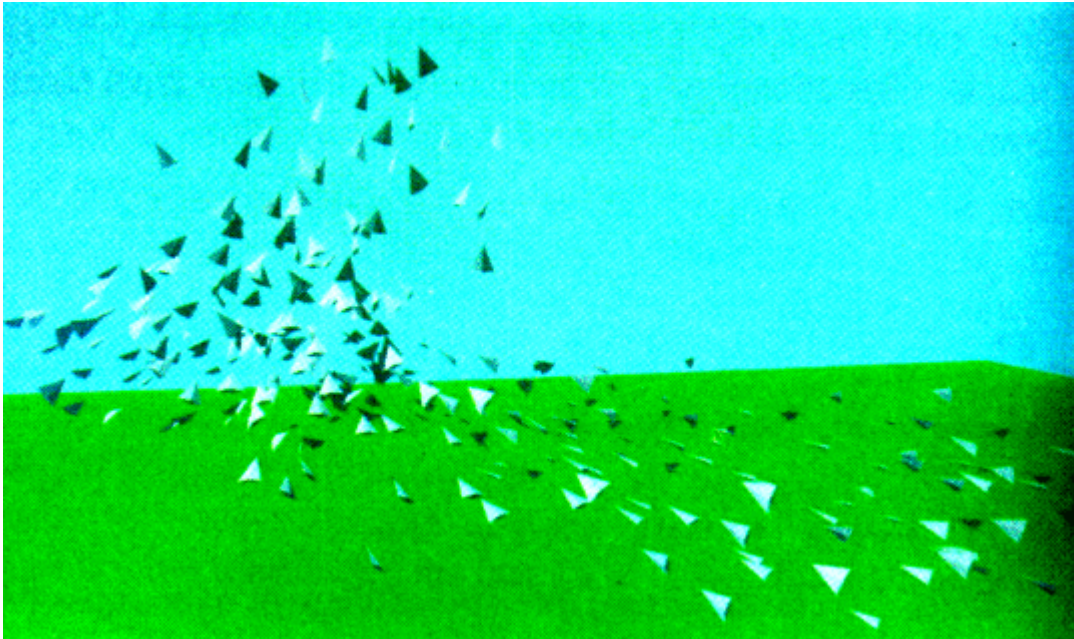
Joonis 6. Boid üritab muuta oma liikumissuunda nii, et see ühtiks naabrite keskmise liikumissuunaga (antud juhul pöörab veidi vasakule). [10]

3. Liikumine lokaalse parve keskkoha suunas:



Joonis 7. Boidi kõik naabrid asuvad temast vasakul ja seetõttu hoiab ta ka ise veidi vasakule, et vältida parvest lahkumist. [10]

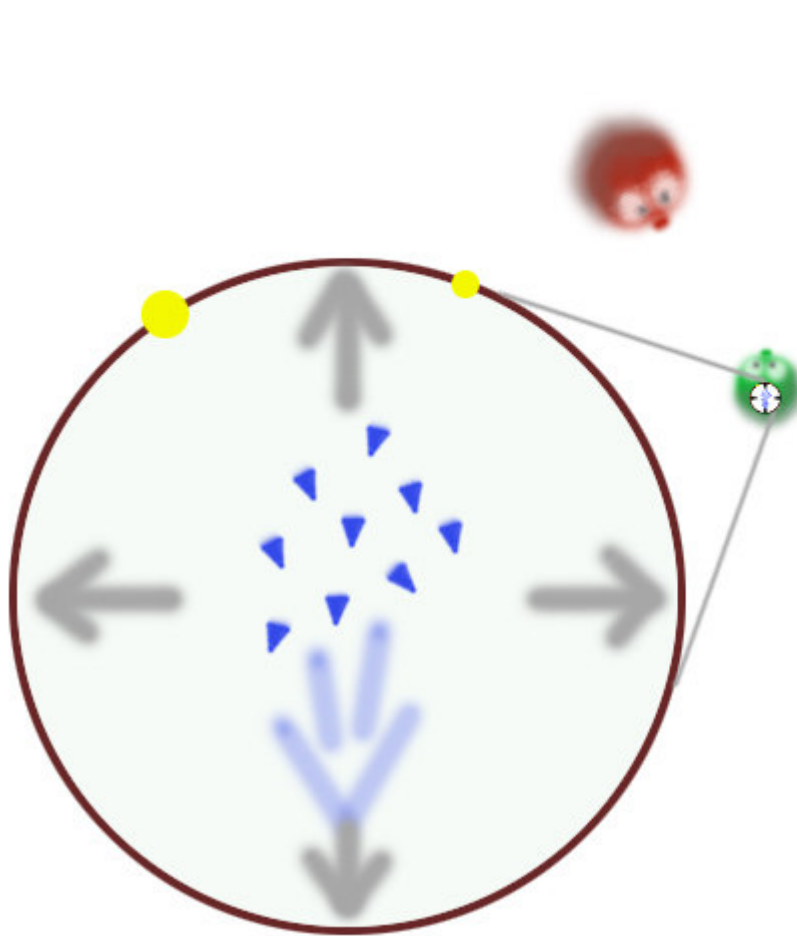
Lisaks tuleks, vähemalt veidigi realistliku simulatsiooni saavutamiseks, modelleerida boidide ja nende keskkonna mõningaid füüsikalisi omadusi nagu mass ja takistusjõud, vältimaks indiviidide ülisuuri kiirusi ja kiirendusi. Kui võtta aluseks lihtne füüsikaline mudel ja eelnimetatud reeglid, siis tulemuseks ongi küllaltki usutavana näiv parvekäitumine (joonis 8).



Joonis 8. Boidimudeli põhjal loodud parvekäitumise simulatsioon. [13]

Minu idee on asetada simuleeritud parv virtuaalsesse suletud ruumi, lisada boididele mõned sensorid (näiteks valguse tugevust ja suunda tajuvad) ja asetada ruumi neile sensoreile vastavad allikad (näiteks siis valgusallikad). Peale selle tuleks boididele juurde kirjutada mõned reeglid, mis mõjutavad nende liikumist sõltuvalt sensoritelt tulevatest andmetest. Et selline suletud ruumis asuvast parvest ja signaallikatest koosnev süsteem moodustaks (potentsiaalse) probleemilahendus- ja/või juhtimissüsteemi, on vaja veel luua sisend- ja väljundkanalid, mille kaudu selle "ajuga" suhelda. Sisendid on võimalik seostada (valgus)allikate parameetritega: asukoht, intensiivsus, värvus, ... Väljunditena saab lugeda mõningaid parve olekut iseloomustavaid muutujaid: parve keskpunkti asukoht ja liikumiskiirus, parve tihedus jms.

Parema ettekujutuse saamiseks süsteemi potentsiaalset konstrueerisin lihtsa mõttelise näite "aju" kasutamisest juhtimissüsteemina mittekriitilise rakenduse (näiteks arvutimängu) jaoks. Eesmärgiks olgu hoida kahemõõtmelises ruumis liikuv objekt (virtuaalne olend arvutimängus) eemal teistest lähedalasuvatest objektidest ("pahadest" tegelastest) {sellest ka idee nimetus "Flockheadz" – parv-ajud(ega elukad)}. Selleks asetame parve ringikujulisse ruumi (joonis 9).



Joonis 9. Lihtne juhtimissüsteem vaenlastest eemale hoidmiseks. Paremäl üleval on toodud situatsioon, milles juhitud objekt (roheline) asub. Suur ring vasakul on virtuaalne ruum, milles liigub parv (väikesed sinised kolmnurgad). Ringi servas on vaenlasi kujutavad valgusallikad (kollased ringid), millest parve liikmed üritavad eemale liikuda ja kogu parv liigub seega allapoole (sinine suur nool). Kui parv jõuab ringi alla, siis hakkab juhitud objekt samuti allapoole liikuma (neli põhiilmakaarte järgi paigutatud halli noolt ringi sees näitavad parve asukohta ja objektile antava juhttoime vahelist seost).

Juhitud objekt suudab määrata vaenlaste kaugust (või ohtlikkust) ja suunda enda suuna suhtes. See info kantakse antud juhul üsna üheselt üle "ajusse", kus ruumi seinale tekita-takse vaenlaste suunas asuvad valgusallikad, mille intensiivsus sõltub vaenlase kaugu-sest. Boididele on lisatud reegel: eemaldu valgusallikatest (ja mida intensiivsem allikas, seda kiiremini). Juhtsüsteemi väljundina vaadeldakse parve keskpunkti asukohta ruumi keskpunkti suhtes: kui näiteks parv asub allaosas, siis juhitud objektile antakse käsk tagurdada. Kui aga parv asub paremal servas, siis juhitud objekt pöördub paremale, jne.

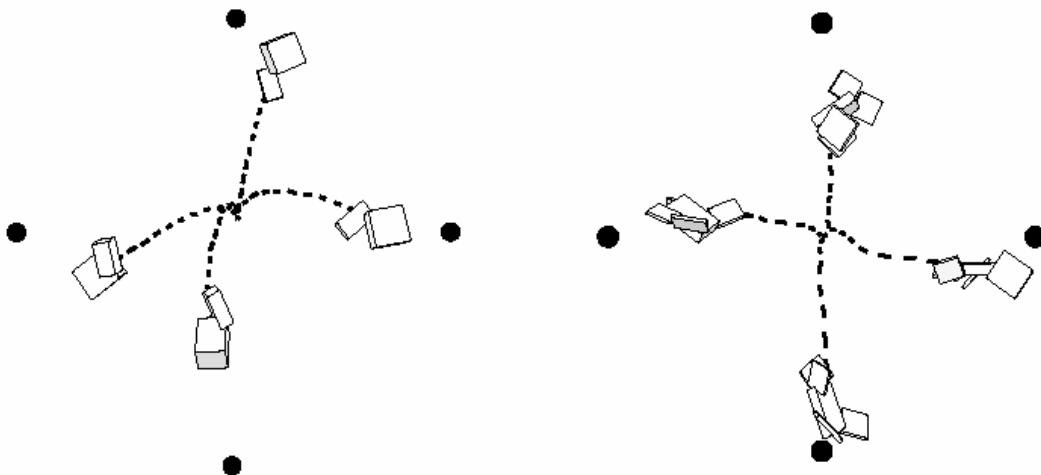
Eelkirjeldatud konstrueeritud süsteem ei ole siiski eriti hea:

- Kui "pahad" objekti ümber piiravad, siis liigub parv ringi keskele ja juhitud objekt ei püüagi põgeneda. Seda probleemi saaks kõrvaldada sobivate reeglite/algoritmide lisa-misega.

- "Pahadest" eemaldumine on otseselt juhtsüsteemi sisse kirjutatud: parve liikmed on disaineri poolt pandud käituma analoogselt juhitavalt objektilt oodatava käitumisega. See tekitab küsimuse juhtimissüsteemi mõttekuse üle: miks mitte anda vältimisreeglid otse juhitavale objektile? Nii lihtsa süsteemi korral ilmselt eriti suurt vahet ei olekski. Võib-olla ainult muudab parve kaasamine juhtimisse objekti liikumist veidi huvitavamaks ja ootamatumaks, vältides otse objektile rakendatava lihtsa eemaldumisalgoritmi kasutamisel vaatlejale jäävat täieliku determineerituse muljet.

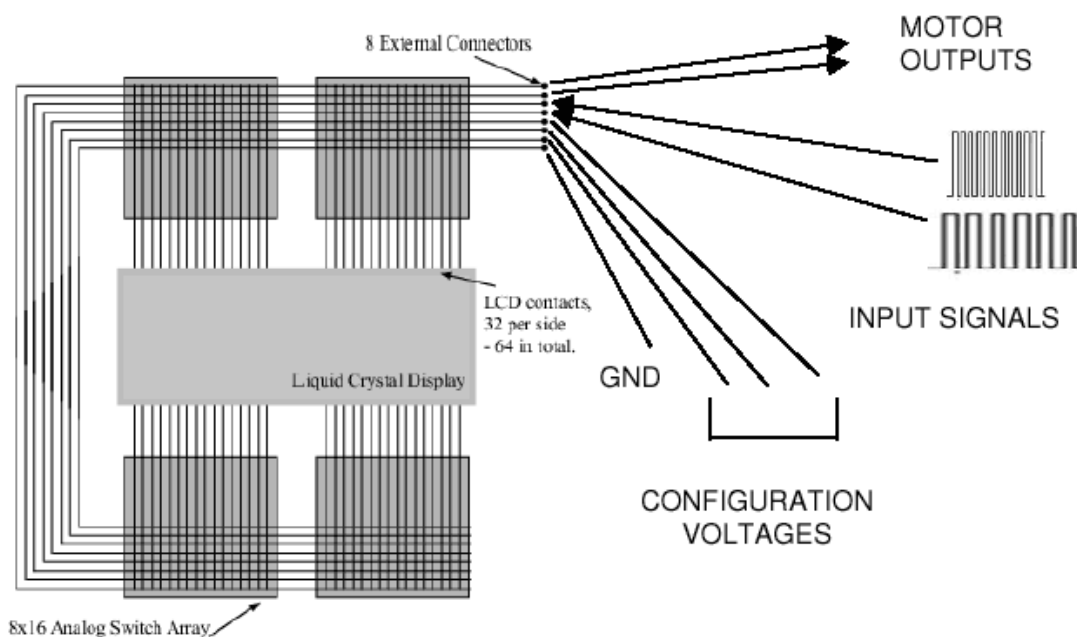
Vähegi keerukamate ja huvitavamalt käitumist tekitavate juhtsüsteemide, mis suudaks toime tulla ka märksa keerukamates olukordades, valmistamine kirjeldatud idee baasil osutub inimdisainerile ilmselt äärmiselt raskeks ülesandeks. Seetõttu tasuks proovida geneetiliste algoritmide rakendamist konkreetsete juhtimisülesannete lahendamiseks (ülesande üldine püstitus ise on virtuaalmaailmades enamasti küllaltki lihtne, näiteks "eesmärk on antud tingimustes ellu jääda" või "eesmärk on liikuda võimalikult kiiresti ühest punktist teise").

Geneetiliste algoritmide käitumine on analoogne looduses toimuva evolutsiooniga: isendeid luuakse geenides asuva info põhjal sobivate ehitusprotsesside poolt, parimad isendid saavad võimaluse oma genee edasi anda, toimuvad geenide rekombineerumised ja mutatsioonid ning kogu seda protsessi korratakse paljude generatsioonide vältel. Geneetilisi algoritme on virtuaalkeskkonnas liikuvatele objektidele juhtimissüsteemide loomiseks suhteliselt "suvalistest" ehitusplokkidest kasutanud näiteks Karl Sims (1994). Sel juhul oli "ajuks" hulk võrguks ühendatud funktsioone (summa, korrutamine, jagamine, min, max, absoluutväärtus, sin, cos, log, integreerimine, tuletis, mälu, ostsillaatorid, jms.), läbi mille sensoritelt tulev info liikus. Geneetiliste parameetritega olid määratud funktsiooniplokkide ühendused omavahel ja sensoritega, plokkides asuvate funktsioonide tüübid ja ühenduste tugevused (kaalud). Lisaks kasutas Sims geneetilisi algoritme ka tehisolendite kehade valmistamiseks. Tulemuseks olid vägagi huvitavate käitumistega isendid:

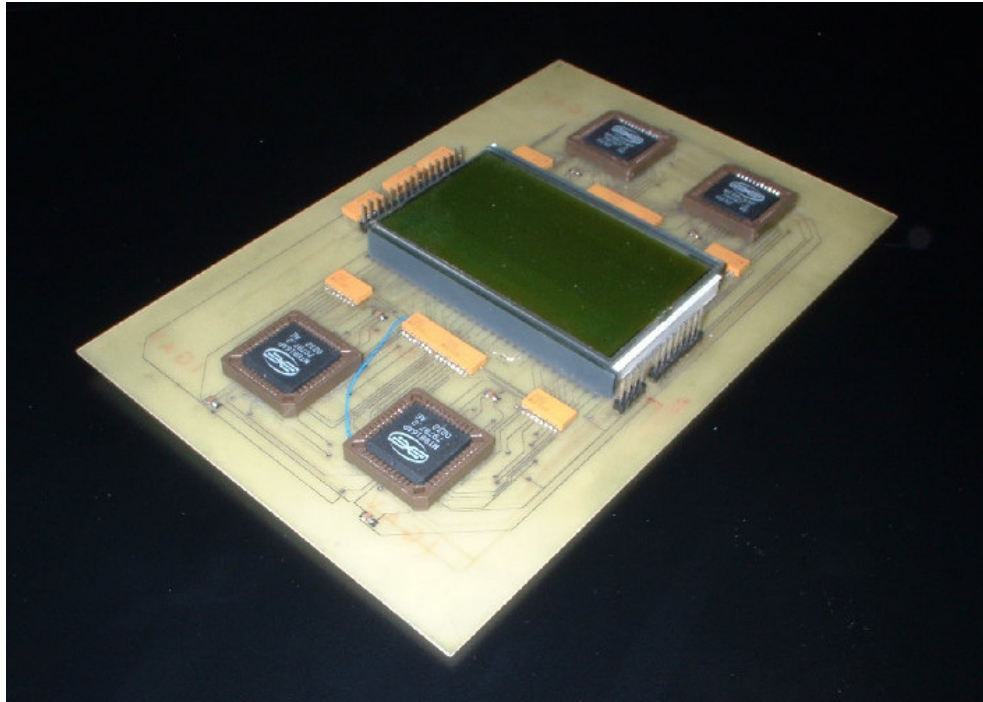


Joonis 10. Kaks Karl Sims'i poolt tehisevolutsiooni abil loodud ujuvat ja valgustäpi suunas liikuvat olendit. Mõlema korral on näidatud neli erinevat katset valgustäpi erinevate asukohtadega. Punktiir tähistab isendi liikumisteed. [14]

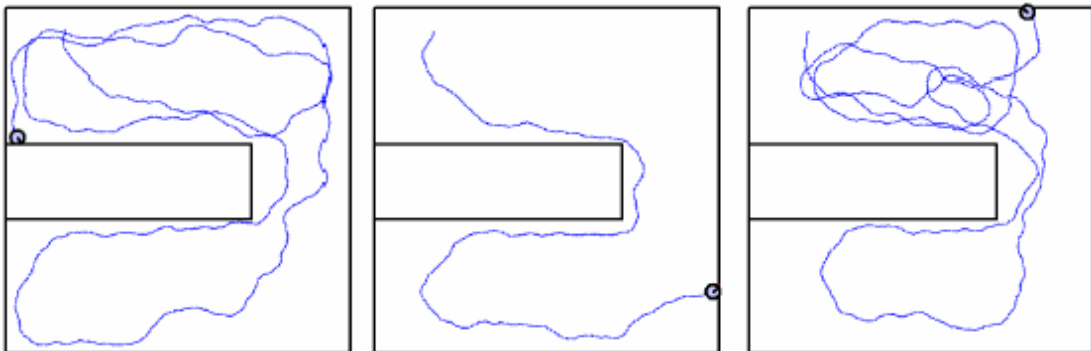
Veelgi ebastandardsemat substraati juhtimissüsteemide valmistamiseks on kasutanud Julian Miller ja Simon Harding. Nad on välja pakkunud idee, et riistvara võiks luua reaalsele füüsilistele materjalidele evolutsiooniliste algoritmide rakendamise teel, kusjuures sobivaks võivad osutuda kõikvõimalikud elektriliselt tundlikud materjalid nagu vedelkristall, nanoosakeste lahused, Langmuir-Blodgett'i kiled (ühe või rohkema molekuli paksused orgaaniliste ainete kihid tahkel pinnal), elektroaktiivsed polümeerid, pingetundlikud kolloidid, kiiritatud räni, mikroobide kolooniad jms. [15]. Oma idee demonstreerimiseks odavate vahendite abil võtsid nad täiesti tavalise taskuarvuti maatriks-tüüpi vedelkristallekraani ja, pärast vajaliku riistvara lisamist personaalarvuti abil signaalide suunamiseks ekraani erinevatele elektroodidele, lasid süsteemil "arenda" näiteks lihtsa simuleeritud roboti "ajuks", nagu seletatud järgnevatel joonistel.



Joonis 11. Skeem väikesest LCD-ekraanist roboti "aju" aretamiseks. Vasakul keskel asub 180x120 pikseline taskuarvuti LCD. Selle 64 elektroodi külge on ühendatud 4 analoogkommutaatorit, mille abil saab mistahes elektroodidega ühendada kaheksat välist sisendit-väljundit (üleval paremal): elektriline "maa(ndus)", kaks (arvuti poolt simuleeritud) sisendsignaali roboti ettesuunatud ultrahelianduritelt, kaks väljundit roboti kummagi ratta mootorite juhtimiseks, ja ülejäänud kolmele juhtmele rakendatakse konstantsed juhtpinged. Geneetilise algoritmi ülesanne on leida, milliste LCD elektroodidega välist sisendit-väljundit ühendada ning millised väärtused anda kolmele juhtpingeklemmidele. [15]



Joonis 12. LCD'le geneetilise algoritmi rakendamise skeemi realisatsioon. [15]



Joonis 13. Evolutsioonilise algoritmi abil loodud füüsilise "juht-LCD" poolt suunatud virtuaalse roboti liikumine. [15]

Selliste huvitavate ja ebatraditsionaalsete näidete edukus annab lootust, et ka Flock-headz'i idee korral võib evolutsioonilise algoritmi kasutamine anda huvitavaid tulemusi. Algoritmile "läbiuurimiseks" antavateks muutujateks sobivad ruumi asetatavate allikate parameetrite sõltuvusfunktsioonid "aju" sisendmuutujatest, väljundite sõltuvusfunktsioonid parve parameetritest ja boidide tegevust määravad funktsioonid. Keerukamate ülesannete lahendamisel võib vajalikuks osutuda süsteemi keerukuse suurendamine: erinevat liiki boidid, kes suhtuvad teistesse boididesse sõltuvalt nende liigist; erinevat liiki sensorid boididel ja vastavad allikad ruumis; sensoritele ja allikatele parameetrite lisamine jne. Alustada tuleks idee uurimist aga siiski võimalikult lihtsa süsteemi loomisest, näiteks eelkirjeldatud konstrueeritud näite algülesande lahendamisest evolutsiooniliselt. Seejuures tuleb muidugi mees pidada, et kui süsteem on liiga lihtne, siis võib selle abil huvitava käitumise saavutamise lihtsaks võimatuks osutada ja süsteemi tuleb nimetatud (või muudel) viisidel täiendada, mitte ideed kohe kõrvale heita.

Kokkuvõte

Tehisintellekti valdkond on veel kaugel sellest, et oma eesmärgid täidetuks lugeda. Toimub pidev arendustöö ja katsetatakse erinevaid lähenemisviise alates otsesest bioloogilistes ajudes toimuvate protsesside modelleerimisest kuni kõikvõimalikke erinevaid ideid ja valdkondi kokkusulatavate mõtete väljapakkumiseni. Käesolevas töös väljapakutud Flockheadz'ide idee kaldub pigem sellesse teise kategooriasse, kuid kasutab siiski olulise komponendina looduslike parvede modelleerimisel leitud põhimõtteid.

Lähitulevikus on mul plaanis idee "Flockheadz" mõistlikkust põhjalikumalt uurida, luues arvutis vastav realisatsioon ja eksperimenteerides sellega, püüdes näiteks tekitada juhtsüsteemi mõne mittekriitilise, esialgu ilmselt meelelahutusliku, rakenduse jaoks.

Viited

- 1. Artificial intelligence.**
Wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Artificial_Intelligence
- 2. Intelligence (trait).**
Wikipedia, the free encyclopedia.
[http://en.wikipedia.org/wiki/Intelligence_\(trait\)](http://en.wikipedia.org/wiki/Intelligence_(trait))
- 3. Neats vs. scruffies.**
Wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Neats_vs._scruffies
- 4. Swarm intelligence.**
Wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Swarm_intelligence
- 5. Ant colony optimization.**
Wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Ant_colony_optimization
- 6. NetLogo Home Page.** Uri Wilensky.
<http://ccl.northwestern.edu/netlogo/>
- 7. Particle swarm optimization.**
Wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Particle_swarm_optimization
- 8. Particle Swarm Optimization.** Christian Jacob and Namrata Khemka.
Evolutionary and Swarm Design Group, 2004.
<http://pages.cpsc.ucalgary.ca/~khemka/pso/model.html>
- 9. A particle swarm model for tracking multiple peaks in a dynamic environment using speciation.** Parrott, D., Xiaodong Li.
Congress on Evolutionary Computation, 19-23 June 2004, Vol. 1, pp. 98-103.
<http://ieeexplore.ieee.org>
- 10. Boids (Flocks, Herds, and Schools: a Distributed Behavioral Model).**
Craig Reynolds.
<http://www.red3d.com/cwr/boids/>
- 11. Boids.** Conrad Parker.
<http://www.vergenet.net/~conrad/boids/>
- 12. Cool School.** David S. Hooper.
<http://www.kewlschool.com/>
- 13. Flocks, Herds, and Schools: A Distributed Behavioral Model.** Reynolds, C. W.
Computer Graphics, 21(4) (SIGGRAPH '87 Conference Proceedings), pp. 25-34.
<http://www.red3d.com/cwr/papers/1987/boids.html>

14. Evolving Virtual Creatures. Karl Sims.

Computer Graphics, Annual Conference Series, (SIGGRAPH '94 Proceedings),
July 1994, pp.15-22.

<http://www.genarts.com/karl/>

15. Evolution In Materio: Programming materials to perform computation using evolution. Simon Harding.

<http://www.evolutioninmaterio.com/slides.html>