SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Stohastička difuzijska pretraga

Vedran Šuman

Voditelj: doc. Dr. Sc. Marin Golub

Zagreb, studeni, 2008. godina

Sadržaj

1. Uvod

Cilj ovog dok dokumenta je pojasniti Stohastičku difuzijsku pretragu (eng. *Stochastic diffusion search*), u daljnjem tekstu SDS.

SDS je algoritam iz obitelji Evolucijskih algoritama koji se primarno koristi za probleme optimizacije, a može se svrstati u potkategoriju *Swarm Inteligence* (inteligencija roja).

U okviru ovog rada napraviti će se kratki teorijski uvod u problematiku, te će se pobliže opisati sam algoritam i pojasniti njegov pseudokod. Prikazati će se jedan problem na koji se može primijeniti SDS i njegova usporedba s nekoliko drugih algoritama.

U praktičnom dijelu biti će opisano programsko rješenje za problem pretrage u kojem je korišten SDS.

2. Stohastička difuzijska pretraga

2.1 Uvod u problematiku

SDS je prvi puta opisao Dr. John Mark Bishop 1989. godine. SDS je efektivna generička metoda pretraživanja, originalno razvijena kao populacijski orijentiran algoritam koji traži odgovarajuće uzorke. Algoritam je dio Inteligencije Roja (eng. Swarm Intelligence) i algoritama optimizacije i pretraživanja, inspiriranih socijalnim insektima (pčele i mravi), kao što su Optimizacija Kolonijom Mrava *(eng. Ant Colony Optimization,* u daljnjem tekstu ACO), Optimizacija Partikularnim Rojem (eng. *Parcticle Swarm Optimization,* u daljnjem tekstu PSO) i Genetskim Algoritmima. Zadnjih godina je iskazana velika zainteresiranost za distribuiranim izračunavanjem korištenjem interakcije između agenata.

ACO je zasnovan na komunikaciji koja se zasniva na izmjenama fizičkih osobina simulirane okoline (mravi koji koriste feromone u svojoj okolini). Za takav oblik indirektne komunikacije rabi se izraz Stigmetrička komunikacija (eng. *Stigmetric Communication*). SDS koristi oblik direktne (jedan na jedan) komunikacije između agenata, slične mehanizmu izravne komunikacije prisutne kod jedne vrste mrava, *Leptothorax acervorum.*

SDS koristi populaciju agenata gdje svaki ima hipotezu o mogućem rješenju i evaluira je parcijalno. Agenti izvode jeftinu, djelomičnu procjenu hipoteze (kandidata za rješenja problema pretrage). Potom dijele informacije o hipotezi (difuzija informacija) direktnom jedan na jedan komunikacijom. Kao rezultat difuzijskog mehanizma, veoma kvalitetna rješenja mogu se identificirati pomoću skupine agenata s istom hipotezom. Rad SDS-a je najjednostavnije prikazati Igrom restorana.

SDS je najučinkovitiji za probleme optimizacije gdje se objektna funkcija može rastaviti na dijelove koji se mogu ocijeniti samostalno. Da bi pronašao optimum dane objektne funkcije SDS upošljava roj od n agenata, gdje svaki održava hipotezu o optimumu. SDS algoritam nameće ponavljanje faza TEST (testiraj) i DIFFUSION (podijeli) dok roj agenata ne konvergira optimalnoj hipotezi.

2.1.1 Igra restorana

Grupa izaslanika je nazočna na dugoj konferenciji u njima nepoznatom gradu. Svaku večer moraju pronaći gdje će večerati. Postoji veliki odabir restorana, svaki s velikim odabirom jela. Problem s kojim se grupa suočava je pronalaženje najboljeg restorana gdje će najveći broj delegata uživati u večeri. Paralelna pretraga restorana i kombinacija jela bi bila iscrpljujuća te, najvažnije, preduga. U rješavanju problema Delegati odlučuju primijeniti SDS. [1]

Svaki izaslanik se ponaša kao agent održavajući hipotezu najboljeg restorana u gradu. Svake noći svaki izaslanik testira svoju hipotezu večerajući u svom, trenutno najboljem restoranu neko nasumce odabrano jelo. Iduće jutro, za doručkom svaki izaslanik koji nije uživao u jelu prethodne večeri, pita jednog odabranog kolegu kakva je bila njegova večera. Ukoliko je iskustvo pozitivno i on prihvaća taj restoran za svoj omiljeni, odnosno najbolji. U suprotnom jednostavno izabere nasumce restoran iz telefonskog imenika. Korištenjem ove strategije uočeno je da se vrlo brzo značajan broj izaslanika skupi oko najboljeg restorana u gradu. Taj proces se može opisati idućim algoritmom:

FAZA INICIJALIZACIJE

GDJE SVI AGENTI (IZASLANICI) GENERIRAJU

INICIJALNU HIPOTEZU (RESTORAN)

PETLJA

FAZA TESTIRANJA

SVAKI AGENAT EVAULIRA DOKAZE SVOJE HIPOTEZE (DEGUSTACIJA OBROKA). AGENTI SE DIJELE U AKTIVNE (ONI KOJI SU ZADOVOLJNI VEČEROM) I INAKTIVNE (ONI KOJI NISU).

FAZA DIFUZIJE

INAKTIVNI AGENTI USVAJAJU NOVE HIPOTEZE KOMUNIKACIJOM S DRUGIM AGENTOM, ILI AKO JE AGENT KOJEM SU SE OBRATILI ZA POMOĆ TAKOĐER INAKTIVAN, NEMA RAZMJENE INFORMACIJA, PA NASUMICE USVAJAJU NOVU HIPOTEZU

KRAJ PETLJE

Iteraciom kroz faze testiranja i difuzije agenti stohastički istražuju cijeli prostor rješenja. Agenti provode više vremena oko dobrih rješenja. Pošto svaki agent provodi vrijeme istražujući dobro rješenje, zapošljavajući nove agente, koji zapošljavaju još više novih, prostor s lošim rješenjima se jako malo istražuje. Kandidati za rješenje se identificiraju po velikom broju agenata koji će se okupiti oko njih.

Jaka strana SDS je njegova mogućnost da pobjegne lokalnim minimumima. To je postignuto vjerojatnosnim ishodom parcijalne evaluacije hipoteze u kombinaciji s relokacijom agenata putem stohastičkih mehanizama aktivacije, odnosno novačenja. Parcijalna evaluacija hipoteze dozvoljava agentu brzo formiranje svog mišljenja o hipotezi koju trenutno istražuje bez dugotrajnog testiranja (npr. u igri restorana, agenti mogu pronaći najbolji restoran u gradu bez da probaju jelo u svakom restoranu).

2.1.2 Terminologija

U originalnoj definiciji SDS-a populacija agenata traži najbolje rješenje za problem optimizacije. Skup svih mogućih rješenja problema formiraju prostor rješenja *S*. Svaka točka prostora *S* ima pridruženu objektnu vrijednost. Objektne vrijednosti iz cijelog prostora rješenja S formiraju objektnu funkciju *f*. Zbog pojednostavljenja, pretpostavlja se da je problem minimalizirati sumu od $n\{0,1\}$ - vrijednosti komponente funkcije f_i, koja može biti deterministička:

$$\min_{\forall s \in S} f(s) = \min_{\forall s \in S} \sum_{i=1}^{n} f_i(s) \qquad f_i : S \to \{0,1\} \quad (1.1)$$

Veliki broj problema optimizacije nakon transformacije se rješavaju jednadžbom (1.1), unatoč činjenici da ona postavlja veliko ograničenje. Primjer takvog ograničenja biti će prikazan u poglavlju Primjeri primjene [2.3]. Tijekom rješavanja problema, svaki agent održava hipotezu o najboljem rješenju, odnosno hipotezu koja je kandidat za rješenje, ili označava vrijednost, odnosno točku u prostoru rješenja.

2.2 Algoritam

Originalno agenti u SDS-u rade paralelno i sinkronizirani su. Prolaze kroz razne faze, koje se mogu sažeti u idući pseudokod [2]:

```
INITIALISE (agents);
REPEAT
        TEST (agents);
        DIFFUSE (agents);
UNTIL (halting criterion);
```

INITIALISE: (Inicijaliziraj) Tipično se početna hipoteza svakog agenta bira podjednako nasumce kroz prostor pretrage. Dakako, svaka informacija o vjerojatnom rješenju problema dostupna apriori može biti korištena u postavljanju inicijalne hipoteze. Postoji puno metoda postavljanja hipoteza, ali njihove detaljno poznavanje nije potrebno za razumijevanje samog algoritma.

TEST: (Testiraj) Svaki agent nasumice izabire jednu komponentu funkcije f_i, $i \in \{1,...,n\}$, i evaluira partikularnu hipotezu $s_h \in S$. Ovisno o ishodu hipoteze, agenti se dijele u dvije skupine: aktivne i neaktivne. Za aktivne agente vrijedi $f_i(s_h) = 0$, a za neaktivne agente vrijedi $f_i(s_h) = 1$. Testnu fazu moguće je opisati idućim pseudokodom:

```
FOR AGENT = 1 TO (ALL AGENTS)
        KOMPONETNTA FUNCKIJE = ODABERI-NASUMICE-KOMPONENTU-FUNKCIJE()
        IF( KF( AGENT.HIPOTEZA) == 0)  AGENT.AKTIVNOST = ISTINA;
                ELSE
                AGENT.AKTIVNOST = NEISTINA;
                END
END
```

Booleanova testna funkcija vraća ISTINA ako je nasumce odabrana djelomična procjena objektne funkcije pokazatelj dobre hipoteze.

DIFFUSE: (Difuzija, Razlaganje) Tijekom ove faze, svaki neaktivni agent nasumice odabire agenta za komunikaciju. Ukoliko je odabrani agenta aktivan, onda prvi agent preuzima njegovu hipotezu: difuzija informacija. Ukoliko je odabrani agenta neaktivan, nema hipoteze, nema razmjene podataka, pa prvi agent nasumice odabire neku hipotezu. U standardnom SDS-u aktivni agenti ne započinju komunikaciju, već samo neaktivni. Ova faza se može opisati idućim pseudokodom:

```
FOR AGENT 1 TO (ALL AGENTS)
    IF (AGENT.AKTIVNOST == NEISTINA)
        AGENT2 = ODABERI-NASUMICE-AGENTA(AGENTI);
        IF (AGENT2.AKTIVNOST == ISTINA)
            AGENT.HIPOTEZA = AGENT2.HIPOTEZA
            ELSE
            AGENT.HIPOTEZA = ODABERI-NASUMICE-HIPOTEZU();
            END
        END
END
```

KRITERIJ ZAUSTAVLJANJA (Halting criterion) Postoji puno kriterija zaustavljanja. Njihovo detaljno poznavanje također nije potrebno za razumijevanje algoritma. Za otkrivanje hipoteze koriste se dva kriterija zaustavljanja: a) *Strong Halting Criterion* koji, nakon što “shvati” da je najveća nakupina agenata premašila minimalan prag, provjerava (stohastički) da li je veličina nakupine stabilna u određenom broju ponavljanja, te b) *Weak Haltering Criterion* koji provjerava stabilnost i minimalan broj aktivnih agenata (ukupna aktivnost uvelike ovisi o najboljem rješenju pronađenom dosad).

2.2.1 Od operacije agenta do ponašanja populacije

Algoritamski opis operacija agenata nije dovoljan da se u potpunosti razumije kako SDS rješavanja probleme optimizacije. Potrebno je razmotriti što se dešava sa populacijom kao cjelinom. Iteraciom faza testiraj i difuzija agenti kao jedinke istražuju cijeli prostor rješenja. Kako faza testiranja češće uspijeva u prostoru s dobrim objektnim vrijednostima, agenti će u prosjeku više vremena provoditi istražujući kvalitetna rješenja, istodobno privlačeći druge agente, koji će pak povlačiti još agenata. To je mehanizam koji uzrokuje formiranje dinamičkih, ali stabilnih nakupina agenata u određenom prostoru rješenja. Ograničenost broja agenata osigurava da samo najbolje rješenje otkriveno do tog trenutka održi stabilan broj agenata u nakupini. Upravo ova neproporcijalna uporaba resursa (broja agenata) na kraju dozvoljava da optimalno rješenje bude pronađeno najvećom nakupinom agenata, bez da svaki agent pretraži cijeli prostor rješenja.

2.2.2 Manipuliranje procesima distribucije resursa

2.2.2.1 Naglasak na lokalno istraživanje

Standardni SDS nema mehanizama za istraživanje sličnosti u samoj objektnoj funkciji. To znači da prostor rješenja često može imati jako slične objektne vrijednosti. Mehanizam uvođenja malih varijacija na različitosti hipoteza se lako može uvesti u postojeći algoritam, što je već učinjeno u rojevima. Jedna mogućnost je narušavanje kopiranja parametara hipoteze uvođenjem nasumičnog pomaka prilikom kopiranja hipoteze za vrijeme faze difuzije. Nešto poput mutacije kod evolucijskih algoritama. Učinak je raspršenje agenata po susjednim lokacijama, umjesto da se samo okupljaju u nakupine oko jedne vrijednosti. To omogućava poboljšano vrijeme konvergencije u prostoru sa sličnostima objektne funkcije i praćenje pokretnih maksimuma dinamičkih problema.

2.2.2.2 Naglasak na globalno istraživanje

Spor zahtjeva za pretraživanjem velikog ili većine prostora rješenja, pogotovo u dinamičkim okruženjima, i potrebe za stabilnim nakupinama koje istražuju trenutno najbolje rješenje ne daje uvijek optimalna rješenja prilikom rabljenja standardnog SDS-a. Njegova distribucija resursa je pohlepna. Nakon što je dobro rješenje otkriveno veliki dio roja će se distribuirati u njegovo istraživanje, što će omogućiti tim agentima da istražuju dalje. Potreban je mehanizam koji će osloboditi neke od tih

agenata, a da neće uvelike narušiti stabilnost agenata u nakupini dobrog rješenja. Takav mehanizam bi povećao učinak SDS-a u velikom broju problema, posebno prilikom dinamičke optimizacije. Jedan takav mehanizam je kontekstno-osjetljiv SDS. Njegova razlika od standardnog SDS-a je u fazi difuzije za aktivne agente. Aktivni agenti standardnog SDS-a cijelo vrijeme održavaju svoju trenutnu hipotezu. Kontekstno-ovisan SDS svakom aktivnom agentu odabire drugog aktivnog agenta za komunikaciju. Ukoliko oba podupiru isto hipotezu, onda prvi agent postaje neaktivan i odabire nasumice novu hipotezu iz prostora rješenja. Mehanizam regulira samog sebe protiv stvaranja ogromnih nakupina agenata, jer što je više agenata s istim hipotezama, veća je vjerojatnost da će upravo oni međusobno komunicirati. Time je uveden mehanizam negativne selekcije u originalni algoritam koji omogućuje SDS-u da održi relativno velike nakupine na višestruko sličnim, skoro optimalnim rješenjima.

2.2.3 Paralelnost SDS-a

SDS je sam po sebi paralelan, no praktično paralelno izvođenje SDS-a je problematično zbog a) potrebe svakog agenta da komunicira s drugima b) količine komunikacije između agenata. Moguće rješenje je organizacija agenata u rešetkastu strukturu s direktnom komunikacijom sa samo *k* najbližih agenata. Drugo rješenje je podjela roja agenata u manje rojeve, gdje se svaki roj izvodi na zasebnom procesoru ali sa potpunom povezanošću uz nisko frekventnu komunikaciju među rojevima.

2.3 Primjeri primjene

2.3.1 Usklađivanje poravnavanjem manje fotografije u veću

Problem je uskladiti poravnavanjem manju fotografiju, koja je slikana iz blago drugačijeg kuta gledanja, sa velikom slikom. Treba napomenuti da je primjer odabran da pokaže princip rada SDS, a ne kao rješenje specifičnog problema. Primjer je prikazan na slici.

Slika 2.3-1 Fotografija za razradu problema

Problem se sastoji od prepoznavanja manje fotografije unutar veće, tako što će se naći koordinate (x,y) koje će dati najbolju podudarnost male fotografije i dijeli velike fotografije. Veličina velike slike je 300 sa 860 piksela, dok je veličina male slike 30 sa 40 piksela. Slika je u RGB formatu, odnosno svaki piksel može imati tri vrijednosti intenziteta boje. Prostor pretrage je diskretan.

Za utvrđivanje podudaranja između dvaju fotografija koristi se matematička funkcija za mjerenje udaljenosti između dvaju točaka, samo što funkcija uspoređuje vrijednosti intenziteta crvene, zelene i plave boje:

$$f(x,y) = \sum_{k,l} (|\, r_{kl} - R_{kl}(x,y)\,| + |g_{kl} - G_{kl}(x,y)\,| + |\, b_{kl} - B_{kl}(x,y)\,|) \quad (1.2)$$

gdje r_{kl} označava vrijednost intenziteta crvene boje piksela (k,l) male fotografije. R_{kl}

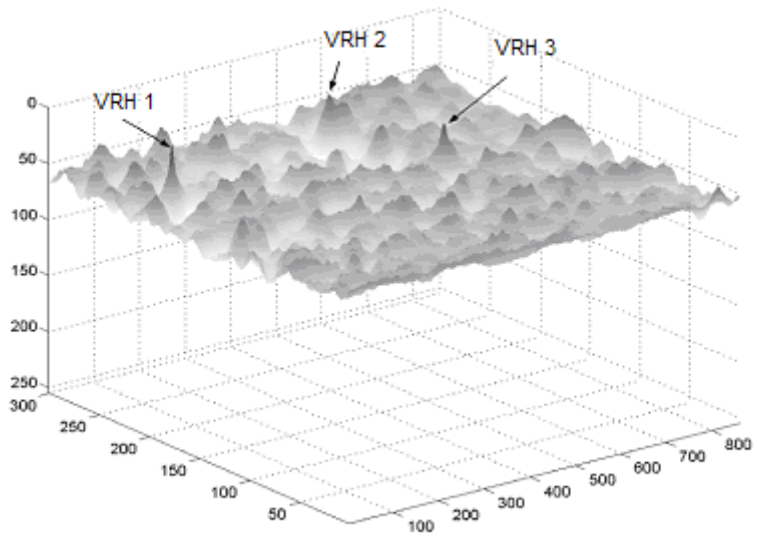
označava vrijednost intenziteta crvene boje piksela (x+k, y+l) u velikoj fotografiji. Problem podudaranja se svodi na rješavanja problema:

$$\min_{x,y} f(x,y) \quad (1.3)$$

Treba uočiti da se objektna funkcija (1.2) može razložiti u više komponentnih funkcija, koje mogu neovisno odrađivati svoj dio procjene (npr. samo za zelenu boju). Prostor rješenja *S* je dvodimenzionalan i diskretan gdje su $x \in \{1,2,3,...,860\}$, a $y \in \{1,2,3,...,300\}$. Veličina prostora rješenja je $860 * 300 = 258000$. Broj komponentnih funkcija f_i je utvrđen uvjetima zbrajanja jednadžbe (1.2) i veličinom male fotografije te različitim vrijednostima intenziteta boja $30 * 40 * 3 = 3600$. Komponente funkcije f_i su diskretne s cjelim brojem u vrijednosti [0,255]. Minimizacijski problem (1.3) se svodi na (1.1). Za komponentu *i* i hipotezu rješenja (x,y), testna procedura računa kvantitetu:

$$t_i(x,y) = \frac{f_i(x,y)}{255} \quad (1.4)$$

Testna procedura daje vrijednost 0 s vjerojatnošću $t_i(x,y)$, dok daje vrijednost 1 sa vjerojatnošću $1 - t_i(x,y)$. Ova procedura osigurava da transformacija objektnih vrijednosti osigurava očuvanje redoslijeda tih vrijednosti, što je dovoljan uvjet za točnu optimizaciju objektne funkcije. Rješenje za dani primjer je prikazano na slici.



Slika 2.3-2 Prostor rješenja problema poravnavanja manje fotografije u veću

Vrh 1 je rješenje, dok su vrhovi 2 i 3 bili kandidati za rješenje, no zbog niže kvalitete

od vrha 1 nisu izabrani.

2.3.2 Usporedba s drugim algoritmima

Iako postoje dobro definirani problemi pretrage za uspoređivanje algoritama kao što je DeJongov test, takav eksperiment za prikaz efikasnosti za korištenje SDS-a nije prikazan zbog nedovoljno definiranih uvjeta težine same pretrage. U ovom odjeljku biti će prikazana empirijska ocjena težine pretrage uspoređujući SDS s nekoliko drugih čestih optimizacijskih algoritama: nasumična pretraga (*random search*), *hill climber*, i *Particle Swarm Optimization* (u daljnjem tekstu PSO). Algoritmi se uspoređuju na problemu opisanom u prethodnom odjeljku 2.3.1.

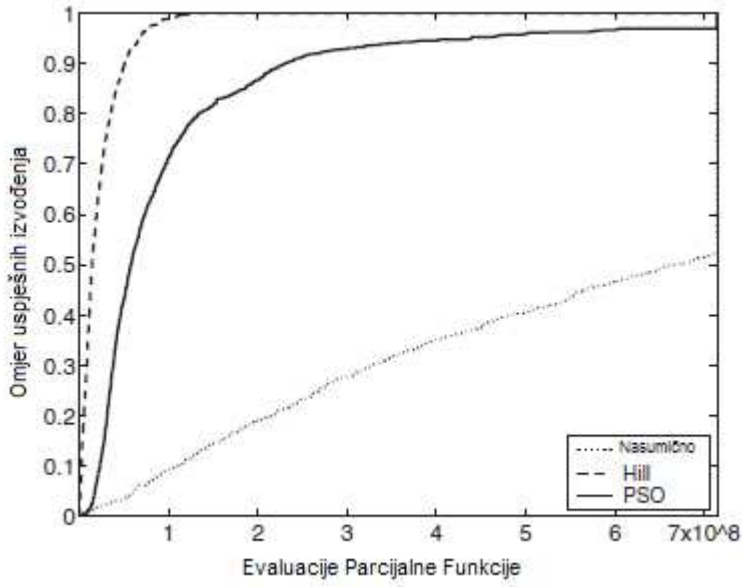
Nasumična pretraga odabire nasumično rješenje i evaluira ga dok ne pronađe Vrh 1 na slici Slika 2.3-2.

Hill climber odabire rješenje nasumično i evaluira ga. U idućim iteracijama 8 susjednih rješenja je evaluirano. Pretraga se pomiče prema najboljem poboljšanju u objektivnoj vrijednosti. Ukoliko takav pomak nije moguć, pretraga je pronašla lokalni optimum, te se ponovno pokreće u novom, nasumično odabranom području. Proces se ponavlja dok se ne otkrije optimalno rješenje, odnosno Vrh 1.

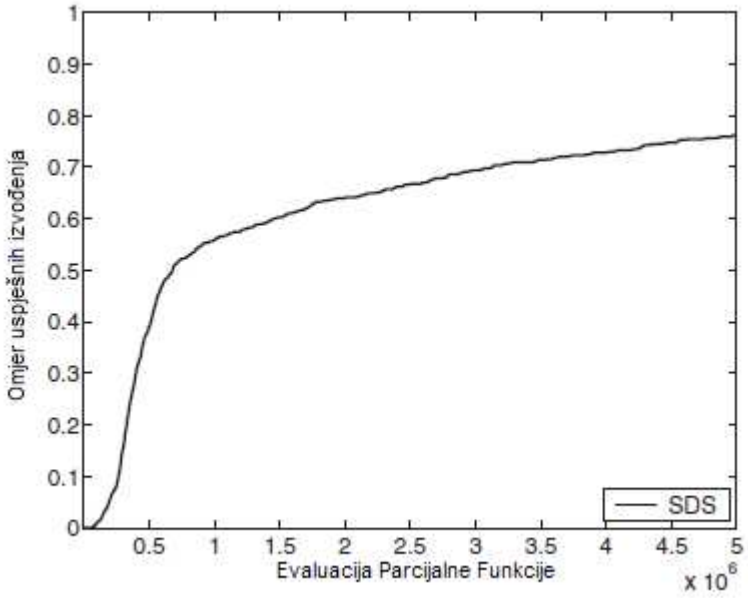
PSO Ovaj algoritam slijedi inačicu PSO ograničenog na lokalne optimume [3]. Algoritam se izvodi dok se optimalno rješenja Vrh 1 ne pronađe s barem jednom česticom, odnosno agentom. Sljedeći parametri su korišteni: koeficijent ograničenja $X = 0.179$, kognitivni i socijalni parametri $c_1 = c_2 = 2.05$, 200 čestica s radijusom susjedstva 1, $\max V_x = 100$ i $\max V_y = 200$. Parametri su odabrani u svrhu davanja optimalnog PSO algoritma pretrage. Za detalje implementacije detalje potražiti u [3].

SDS Korišten je standardni SDS algoritam s 1000 agenata. Mali mutacijski mehanizam opisan u poglavlju 2.2.2 je korišten koji dodaje male pomake (x,y) koordinatama gdje je vjerojatnost mutacije 8%. Pretraga konvergira optimalnom rješenju, tj. Vrhu 1 kad se skupi trećina svih agenata oko njega.

Rezultati izvođenja pokusa su prikazani idućim slikama.



Slika 2.3-3 Usporedba izvođenja Nasumične pretrage, Hill Climbing i PSO. Prikaz za 1000 izvođenja



Slika 2.3-4 Prikaz rezultata 1000 izvođenja algoritma SDS

Slike prikazuju ponašanje četiri gore navedena algoritma prilikom pretrage. Slike prikazuju distribuirano gomilanje ukupnog broja evaluacija parcijalnih funkcija navedenih algoritama. Učinkovitost parcijalne evaluacije može se prikazati usporedbom skupoće, odnosno uloženih resursa, SDS-a i ostalih algoritama.

Nasumična pretraga treba oko 191 000 evaluacija za uspješnost od 50 % u pronalasku globalnog optimuma. To znači da je potrebno 191 000 * 3 600 = 687,6 miliona evaluacija komponente funkcije f_i. SDS će napraviti oko 683 000 komponentnih evaluacija. PSO zahtijeva 16 000 potpunih evaluacija funkcija, a Hill climbing 4 000.

Zanimljivo je pogledati, ne uspoređujući brojeve, sa koliko komponentnih funkcija f_i SDS nadalje nadmašuje druge algoritme. Vjerojatnostni parcijalni evaluacijski mehanizam SDS-a pretvara pretragu u stohastički dinamički proces neovisan o broju komponentnih funkcija u objektnoj funkciji. One ovise samo o obliku prostora pretrage. Drugim riječima, bez obzira da li za prostor pretrage treba 100, 1 000 ili 100 tisuća komponentnih funkcija, prosječno ponašanje pretrage SDS-a je uvijek isto. Konkretno za prostor pretrage prikazan na slici Slika 2.3-1, SDS bi nadmašio nasumičnu pretragu za objektne funkcije koje se sastoje od 683 000 / 191 000 ≈ 4 ili više komponentnih funkcija. Za PSO se dobije 683 000 / 16 000 ≈ 43, a za Hill climbing 683 000 / 4 000 ≈ 171. Relativno loši rezultati PSO-a naspram Hill climbing-a može se objasniti da se roj sastoji od 200 jedinki, odnosno čestica gdje svaka izvodi potpunu parcijalnu evaluaciju. Vjerojatno bi se učinkovitost PSO-a poboljšala ukoliko bi se povećale dimenzije problema.

2.4 SDS i evolucijski algoritmi

SDS algoritam opisan perspektivom agenata koji iteriraju fazu testiraj i difuzija može se učiniti da je daleko, ili da uopće nije evolucijski algoritam. Iako nije originalno potekao iz metafora biološke evolucije, nego promatrajući neuronske mreže, SDS se uklapa u algoritme inspirirane Darwinovom evolucijom izvedenih osnovnim procesima varijacije, selekcije i replikacije. SDS se uklapa iz perspektive hipoteze. SDS čini nasumične odabire i male preinake hipoteza prilikom njihovog kopiranje, kao što evolucijski algoritmi rade uvođenje nasumičnih imigranata i mutacije. Odbacivanje hipoteze je njezina “smrt”, dok je kopiranje hipoteze svojevrsna reprodukcija. Dobre hipoteze imaju veću šansu preživljavanja faze testiranja, te istovremeno se šire na veći broj agenata. Zbog ograničenog broja agenata, samo određeni broj agenata može održavati hipotezu, te kroz međusobnu komunikaciju agenata ostvaruje se selekcija, iako nema klasične selekcije. Zbog kontinuirane i indirektne evaluacije svake hipoteze, SDS simulira evolucijske procese u nešto drugačijem vremenski definiranom prostoru, za razliku klasične linearne evolucije.

2.5 Usporedba SDS i algoritama socijalnih insekta

Nasuprot stigmetričnoj komunikaciji korištenoj u većini algoritama mrava, SDS koristi jedan na jedan sistem novačenja po uzoru na ponašanje određenih vrsta mrava koji ne ostavljaju feromone, nego idu jedan za drugim. Iz veze s SDS-om tvrdi se da se efikasna i globalna odluka može postići samo interakcijom i međusobnom komunikacijom među populacijom, gdje svaka jedinka održava svoju hipotezu i parcijalno je evaluira. Procesi novačenja novih agenata su mnogo kompleksniji, s obzirom na to da agent u SDS-u ne mora fizički otići do drugog agenta (mrava), te izvoditi rutine ponašanja, odnosno 'plesa' kako bi mu prenjeo svoju hipotezu i unovačio ga. Nigdje u prirodi ne postoji sistem novačenju orijentiran na aktivne i neaktivne agente. Sistem je preuzet od određenih vrsta mrava i njihovog ponašanja prilikom osnivanja novog mravinjaka. Promatrajući ih, došlo se do zaključka da ti mravi trebaju veću razinu individualnih kognitivnih sposobnosti, kao što je usporedba dva pogodna mjesta za podizanje novog mravinjaka, da bi došli do optimalnog rješenja. Što je suprotno stigmetričnoj komunikaciji većine mrava. Unatoč tome, osnovne sličnosti SDS-a i socijalnih insekta sugerira da globalna i čvrsta odluka u obadva sistema proizlazi iz kooperacije povezanih agenata, gdje svaki od njih individualno ne bi bio sposoban riješiti problem u približno istom vremenu.

3. Projektiranje SDS sustava

3.1 Ideja

Osnovni cilj ostvarenog programa je pretraga, i uklapanje male slike u veliku, kao što je to opisano u poglavlju 2.3.1. Treba naglasiti da je poglavlju 2.3.1 predviđeno da je mala slika slikana iz blago različitog kuta. U ostvarenju ovog programa je predviđeno da je mala slika dio velike, odnosno njezina potslika. Ulaz programa su dvije slike u bmp formatu. Izlaz je u tekstualnom obliku. Ime programa je Stochastic Diffusion Search.

3.2 Ulazi i izlazi programa

Kao ulazi programa zamišljene su dvije slike u bmp formatu. Predviđeno je da prva slika bude veličine 500*300 pixela, a mala slika 50*30 pixela. Program radi i za slike drugih veličina, koje moraju biti manje od zadanih dimenzija, iako je izvorno predviđen za zadane dimenzije.

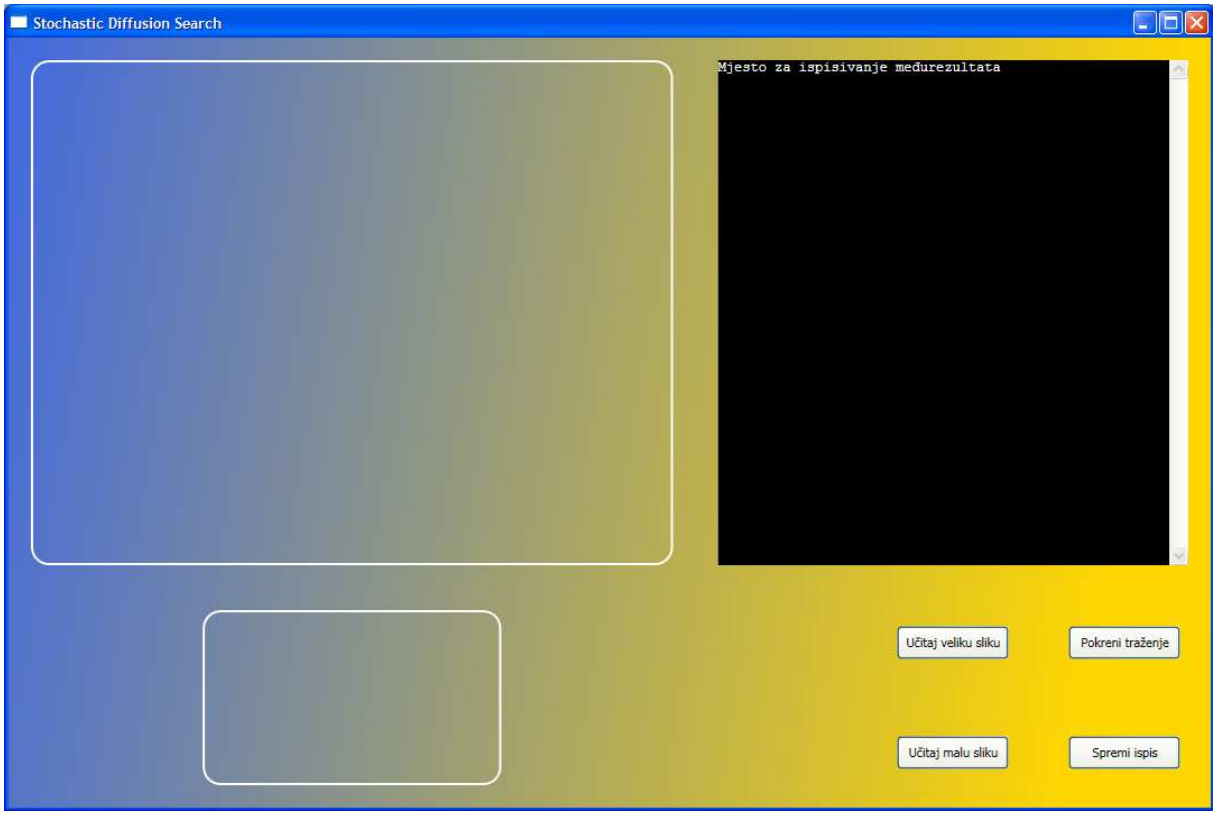
Izlazi programa su na desnoj strani glavnog prozora programa i ispisuju se u obliku koordinata na kojima se agenti trenutno nalaze. Rezultate je moguće spremiti u obliku txt datoteke.

3.3 Realizacija

Program je ostvaren programskim jezikom C# u razvojnom okruženju Microsoft Visual C# 2008 Express Edition. Za pokretanje programa nisu potrebni nikakvi dodatci, iako računalo mora imati instaliran .NET framework 3.0 ili 3.5.

Agenti SDS-a kao algoritma su programski ostvareni kao četiri dretve gdje svaka pretražuje jedan kvadrant slike.

Program se pokreće kao posebna Win32 aplikacija.



Slika 3.3-1: Aplikacijski prozor programa Stochastic Diffusion Search

Korisnik učita slike u program i nakon toga glavni program pokreće 4 neovisne dretve, od kojih svaka pretražuje svoj kvadrant (kao kvadranti u koordinacijskom sustavu).

Agenti rade tako da kreću od početka svog kvadranta, odnosno u gornjem lijevom kutu svog kvadranta. Te uzimaju vrijednost boje tog piksela koju dobivaju kao cjelobrojnu vrijednost. Bitno je napomenuti da ovdje program ne razlaže vrijednosti boja u RGB format kao što je to bio slučaj u poglavlju 2.3.1, već gledaju boju u jednom formatu.

Ukoliko se prva vrijednost boja piksela slažu, agent prelazi na drugi piksel male i velike slike, sve dok ne nađe potpuno preklapanje. Ukoliko nađe poklapanje ispiše rezultat.

Dretve, odnosno agenti, će svakih deset piksela ispisati gdje se nalaze i da li su pronašle rješenje.

3.4 Upute za korištenje

Nakon što korisnik pokrene aplikaciju otvara se prozor kao što je to prikazano na slici Slika 3.3-1. Lijevi dio prozora je rezerviran a prikaz velike i male slike, dok se desno ispisuju rezultati. Na dnu desnog kuta se nalaze četiri gumba za kontroliranje programa.

Prvi gumb je 'Učitaj veliku sliku'. Nakon njegovog pritiskanja korisniku se otvara novi prozor u kojem treba učitati veliku sliku. Slika mora biti bmp formata (inače program neće raditi) i korisnik mora pripaziti na dimenzije kako je već opisano.

Drugi gumb je 'Učitaj malu sliku' i procedura je ista kao za veliku sliku, no ovaj put korisnik treba učitati malu sliku.

Treći gumb je 'Pokreni traženje' čijim se pritiskom pokreće pretraga, odnosno stvaraju se i aktiviraju agenti.

Četvrti gumb je 'Spremi ispis' kojim se svi rezultati koji su prikazani u prozoru rezultata mogu spremiti u obliku txt datoteke. Korisniku će se otvoriti novi prozor gdje može odabrati ime datoteke i njezinu lokaciju. Ukoliko prilikom čitanja dobivene txt datoteke ispis bude nečitak, ili slova budu gusto naslagana, preporuča se otvaranje dobivene txt s Microsoft Wordom koji će od korisnika zatražiti da odabere dekodiranje. Potrebno je odabrati Other encoding, Unicode (UTF 8).

Za ponovnu pretragu, program se mora ugasiti i ponovno pokrenuti. Nije dovoljno samo učitati nove slike i stisnuti gumb 'Pokreni traženje'.

4. Eksperimenti

Program je testiran s proizvoljnim slikama koje su preporučenih dimenzija iz prethodnog poglavlja.

Velika slika za testiranje je prikazan na donjoj slici. Dok je mala slika zbog lakšeg uočavanja istaknuta unutar veće.

Slika 3.4-1 Velika slika za testiranje programa

Nakon izvođenja programa, i spremanja ispisa dobiva se sljedeći rezultat:

```
Inicijalizacija agenata (dretve
pretrazivaci).
Rezultat početni, rješenje još ne
postoji : x=-1, y=-1
Agent je aktiviran.
Agent je aktiviran.
Agent je aktiviran.
Agent je aktiviran.
Trenutna pozicija je : x=250, y=150
Trenutna pozicija je : x=250, y=160
Trenutna pozicija je : x=0, y=0
Trenutna pozicija je : x=250, y=0
Trenutna pozicija je : x=0, y=150
Trenutna pozicija je : x=250, y=170
Trenutna pozicija je : x=0, y=10
Trenutna pozicija je : x=250, y=10
Trenutna pozicija je : x=0, y=160
Trenutna pozicija je : x=250, y=180
Trenutna pozicija je : x=0, y=20
Trenutna pozicija je : x=250, y=20
Trenutna pozicija je : x=0, y=170
Trenutna pozicija je : x=250, y=190
Trenutna pozicija je : x=0, y=30
Trenutna pozicija je : x=250, y=30
```

```
Trenutna pozicija je : x=0, y=180
Trenutna pozicija je : x=250, y=200
Trenutna pozicija je : x=0, y=40
Trenutna pozicija je : x=250, y=40
Trenutna pozicija je : x=0, y=190
Trenutna pozicija je : x=250, y=210
Trenutna pozicija je : x=0, y=50
Trenutna pozicija je : x=250, y=50
Trenutna pozicija je : x=0, y=200
Trenutna pozicija je : x=250, y=220
Trenutna pozicija je : x=0, y=60
Trenutna pozicija je : x=250, y=60
Trenutna pozicija je : x=0, y=210
Trenutna pozicija je : x=0, y=70
Trenutna pozicija je : x=250, y=70
Trenutna pozicija je : x=0, y=220
Trenutna pozicija je : x=250, y=230
Trenutna pozicija je : x=0, y=80
Trenutna pozicija je : x=250, y=80
Trenutna pozicija je : x=0, y=230
Trenutna pozicija je : x=250, y=240
Trenutna pozicija je : x=250, y=250
Trenutna pozicija je : x=0, y=90
Trenutna pozicija je : x=250, y=90
```

```
Trenutna pozicija je : x=0, y=240
Trenutna pozicija je : x=250, y=260
Trenutna pozicija je : x=0, y=100
Trenutna pozicija je : x=250, y=100
Trenutna pozicija je : x=0, y=250
Trenutna pozicija je : x=250, y=270
Trenutna pozicija je : x=0, y=110
Trenutna pozicija je : x=250, y=110
Trenutna pozicija je : x=0, y=260
Trenutna pozicija je : x=0, y=120
Trenutna pozicija je : x=250, y=120
Trenutna pozicija je : x=0, y=270
Trenutna pozicija je : x=0, y=130
Trenutna pozicija je : x=250, y=130
Trenutna pozicija je : x=0, y=140
Trenutna pozicija je : x=250, y=140
Trenutna pozicija je : x=260, y=150
Trenutna pozicija je : x=260, y=160
Trenutna pozicija je : x=10, y=150
Trenutna pozicija je : x=260, y=170
Trenutna pozicija je : x=10, y=160
Trenutna pozicija je : x=260, y=180
Trenutna pozicija je : x=10, y=170
Trenutna pozicija je : x=10, y=180
Trenutna pozicija je : x=260, y=190
Trenutna Pozicija je : x=2600, y=200
Trenutna pozicija je : x=10, y=190
Trenutna pozicija je : x=260, y=210
```

```
Trenutna pozicija je : x=10, y=200
Trenutna pozicija je : x=260, y=220
Trenutna pozicija je : x=10, y=210
Trenutna pozicija je : x=260, y=230
Trenutna pozicija je : x=10, y=220
Trenutna pozicija je : x=260, y=240
Trenutna pozicija je : x=10, y=230
Trenutna pozicija je : x=260, y=250
Trenutna pozicija je : x=10, y=240
Trenutna pozicija je : x=260, y=260
Trenutna pozicija je : x=10, y=250
Trenutna pozicija je : x=260, y=270
Trenutna pozicija je : x=10, y=260
Trenutna pozicija je : x=10, y=270
Trenutna pozicija je : x=260, y=0
Trenutna pozicija je : x=260, y=10
Trenutna pozicija je : x=260, y=20
Trenutna pozicija je : x=260, y=30
Trenutna pozicija je : x=260, y=40
Trenutna pozicija je : x=260, y=50
Trenutna pozicija je : x=260, y=60
Trenutna pozicija je : x=260, y=70
Trenutna pozicija je : x=260, y=80
Trenutna pozicija je : x=260, y=90
Trenutna pozicija je : x=260, y=100
Trenutna pozicija je : x=260, y=110
Trenutna pozicija je : x=260, y=120
Trenutna pozicija je : x=260, y=130
Trenutna pozicija je : x=260, y=140

Rezultat agenta  je : x=9, y=39
```

Dakle može se zamijetiti sa sva četiri agenta, odnosno dretve programa, pretražuju neovisno u svom kvadrantu, te ispisuju svoju poziciju nakon pomaka za 10 piksela. Nakon što jedan agent pronađe rješenje i ispiše ga, on to dojavljuje drugim agentima, koji se nakon toga gase i program završava.

5. Zaključak

SDS je dio inteligencije roja, a nastao je promatrajući socijalne insekte u prirodi. Njegova specifičnost je u komunikaciji agenata jedan na jedan. [2.1] Iz toga proizlazi njegov pseudokod koji se najapstraktnije može opisati u dvije faze: test i difuzija. [2.2] Iteraciom tih procesa vrlo brzo dolazi do nakupljanja agenata oko najboljeg rješenja. Snaga SDS-a je što svaki agent ne mora pretražiti cijeli prostor rješenja. Iako se po želji korisnika to može promijeniti manipulacijom procesa distribucije resursa. [2.2.2]

SDS je sam po sebi paralelan, no postoje poteškoće u ostvarivanju potpunog paralelnog izvođenja.

Pokazano je da je SDS u principu može koristiti za stohastičke i dinamičke optimizacijske probleme. Prikazan je njegov algoritam i mogućnost prilagodbe istog. Njegova mogućnost laganog prilagođavanja između naglaska na lokalni prostor ili sveukupni prostor pretrage prostora rješenja uz njegovu parcijalnu evaluaciju čini ga potencijalno jako korisnim u primjeni rješavanja navedenih problema. Istražuje se njegova primjena u aplikacijama za traženje manjih dijelova sveukupne slike, npr. traženja očiju na licu.

SDS je trenutno aktivan u nekim alatima za pretrage teksta i za trenutnu pretragu Interneta. Također se njegov princip koristi u WLAN mrežama, gdje računalo koje se spaja na usmjeritelj (eng. *router*) ne traži samo koji mu je usmjeritelj najbolji, već tu odluku donosi u komunikaciji s drugim računalima u blizini.

6. Literatura

[1] Prvi izvor: http://www.scholarpedia.org/article/Stochastic_diffusion_search

[2] Drugi izvor: Kris de Meyer, Slawomir J. Nasuto, Mark Bishop: Stochastic Diffusion Search: Partial Function Evaluation in Swarm Intelligence Dynamic Optimisation

[3] Treći izvor: Kennedy, J, Eberhart, R C (2001) Swarm Intelligence. Morgan Kaufmann