Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aleksandar Kojić

Vizualizacija optimizacije z rojem delcev na mobilni platformi

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Marko Robnik-Šikonja

Ljubljana 2012

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Aleksandar Kojić, z vpisno številko **63080216**, sem avtor diplomskega dela z naslovom:

Vizualizacija optimizacije z rojem delcev na mobilni platformi

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Marka Robnik-Šikonje,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 13. septembra 2012 Podpis avtorja:

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00267/2012
Datum: 11.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ALEKSANDAR KOJIĆ**

Naslov: **VIZUALIZACIJA OPTIMIZACIJE Z ROJEM DELCEV NA MOBILNI PLATFORMI**

VISUALIZATION OF PARTICLE SWARM OPTIMIZATION ON MOBILE PLATFORM

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Pri poučevanju hevrističnih optimizacijskih algoritmov potrebujemo orodja za boljšo predstavitev vsebin in za njihovo lažje razumevanje. Preiskovanje z rojem delcev je priljubljena in uspešna optimizacijska tehnika. Njeno delovanje je mnogo lažje razumeti z vizualizacijo na izbranih primerih, ki jih želimo približati študentom z aplikacijo na mobilni oziroma tablični platformi.
Proučite in intuitivno opišite delovanje optimizacije z rojem delcev ter delovanje ilustrirajte na nekaj poučnih problemih, ki omogočajo vizualno prepričljivo predstavitev tudi na omejeni površini mobilne naprave. Na mobilni platformi Android izdelajte animacijo iskanja optimalne rešitve. V mobilno aplikacijo vključite tudi možnost interakcije, ki uporabnika pouči o delovanju in vplivu parametrov ter mu omogoča njihovo nastavitev.

Mentor:
prof. dr. Marko Robnik Šikonja

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Zahvaljujem se mentorju prof. dr. Marku Robnik-Šikonji za vse nasvete in pomoč pri izdelavi diplomske naloge. Zahvaljujem se družini, dekletu Tanji in prijateljem, ki so mi tekom študija stali ob strani.

Kazalo

Povzetek

Namen diplomskega dela je predstavitev optimizacije z rojem delcev na mobilnem operacijskem sistemu Android.

V prvem delu diplomskega dela opišemo zgradbo operacijskega sistema Android in aplikacij na tej platformi. V nadaljevanju predstavimo glavne značilnosti optimizacije z rojem delcev, kje jo lahko najdemo v naravi in kako jo uporabljamo v računalništvu.

Drugi del diplomskega dela je namenjen vizualizaciji optimizacije z rojem delcev. Predstavimo analizo, načrtovanje, izdelavo in testiranje aplikacije.

Ključne besede: optimizacija z rojem delcev, PSO, vizualizacija, Android, aplikacije Android.

Abstract

The aim of the thesis is a presentation of particle swarm optimization on mobile operating system Android.

First we describe the structure of the operating system Android and Android applications. We present the main features of particle swarm optimization, where it is found in the nature and how it is used in computer science.

The second part of the thesis presents visualization of particle swarm optimization. We present an analysis, planning, developement and testing of the application.

Key words: particle swarm optimization, PSO, visualization, Android, Android applications.

Poglavje 1

Uvod

Algoritmom in njihovi optimizaciji smo med študijem namenili precej pozornosti, saj se bomo z njihovim razumevanjem in s pravilno implementacijo pogostokrat srečali, ko bo potrebno reševati probleme. Takšno študijsko snov je težko razumeti, zato smo se odločili za vizualizacijo algoritma, ki naj bo študentom v pomoč pri učenju in razumevanju delovanja algoritma. Ideja algoritma optimizacija z rojem delcev je medsebojno sodelovanje osebkov pri doseganju skupnega cilja v določenem prostoru.

Za vizualizacijo optimizacije sem izbral mobilni operacijski sistem Android, saj je uporaba pametnih mobilnih naprav vedno bolj razširjena in so te aplikacije zelo priljubljene med študentsko populacijo. Eden od razlogov za izbiro je preprost dostop do aplikacije - mobilna naprava je sorazmeroma majhna in večina jo nosi vedno s seboj. Za razvoj v Androidu sem se odločil predvsem zato, ker je aplikacije možno pisati v programskem jeziku java, s katerim smo se dodobra spoznali tekom študija. Mobilni operacijski sistem Android je trenutno najbolj priljubljena mobilna platforma.

V nadaljevanju predstavimo kako smo implementirali vizualizacijo optimizacije z rojem delcev. V 2. poglavju bomo predstavili mobilni operacijski sistem Android. Predstavili bomo zgradbo operacijskega sistema, zgradbo aplikacije v Androidu in razvoj aplikacije v razvojnem okolju Eclipse IDE. Tretje poglavje predstavi algoritem optimizacije z rojem delcev, nekatere naj-

bolj uporabljene strukture in dve vrsti optimizacije, ki smo jih vizualizirali. V 4. poglavju bomo predstavili idejo vizualizacije, analizirali bomo specifikacije in zahteve, ki smo si jih postavili. Predstavili bomo načrtovanje in izdelavo aplikacije ter testiranje. Sledijo sklepne ugotovitve in ideje za izboljšavo.

Poglavje 2

Mobilni operacijski sistem Android

2.1 Predstavitev mobilnega operacijskega sistema Android

Mobilni operacijski sistem Android [1, 2] je odprtokodni operacijski sistem za mobilne naprave, kot so pametni telefoni in tablični računalniki. Razvilo ga je podjetje Android Inc., ki ga je leta 2005 kupilo podjetje Google. Temelji na jedru operacijskega sistema Linux. Mobilni operacijski sistem Android uporablja mnogo proizvajalcev mobilnih naprav, kot so Samsung, HTC, Motorola, Sony, LG. Ker je odprtokoden, si ga vsako podjetje lahko prilagodi, tako da se najbolje prilagaja njihovim mobilnim napravam. Pomembni dogodki pri razvoju mobilnega operacijskega sistema Android:

- 2005 - podjetje Google kupi Android Inc.;
- 2007 - ustanovitev konzorcija Open Handset Alliance (OHA), ki združuje podjetja, ki razvijajo odprtokodne mobilne operacijske sisteme;
- 2008 - izdaja Android 1.0 SDK na mobilni napravi G1, ki jo je izdelalo podjetje HTC;

- 2009 - izdaje operacijskega sistema Android 1.5, 1.6, 2.0, 2.1. Več kot 20 naprav uporablja Android;
- 2010 - izdaja Android 2.2, ki teče na več kot 60 mobilnih napravah;
- 2012 - uporaba na več kot 50% vseh mobilnih naprav na svetu.

Lahko bi rekli, da se zgodba Androida začne leta 2007 ob ustanovitvi konzorcija Open Handset Alliance (OHA), ki ga sestavlja 86 podjetij, med njimi Google, HTC, Intel in ostala podjetja, ki se ukvarjajo z mobilnimi tehnologijami. OHA takrat predstavi prvo beta verzijo Androida. Cilj odprtokodnega mobilnega operacijskega sistema Android je boljši in cenejši mobilni operacijski sistem za končne uporabnike. Zaradi tega cilja je Android doživel že nekaj izboljšav oziroma posodobitev. Glavni namen posodobitev je odpraviti napake starejših različic, dodati nove in boljše funkcionalnosti, ki olajšajo delo končnim uporabnikom in izboljšava uporabniškega vmesnika. Prednosti uporabe mobilnega operacijskega sistema Android [3] so:

- je odprtokoden in tako razvijalcem omogoča lažje in cenejše razvijanje programov,
- je brezplačen,
- je enostaven za uporabo,
- je odziven in omogoča večopravilnost,
- ima uporabniku prijazen uporabniški vmesnik,
- samodejna sinhronizacija z Google storitvami,
- omogoča enostavno nalaganje aplikacij iz trgovine Google Play,
- razvijalcem omogoča enostavno distribucijo svojih aplikacij na spletišču Google Play.

VERZIJA	IME	API	DELEŽ
1.5	Cupcake	3	0.2%
1.6	Donut	4	0.5%
2.1	Eclair	7	4.7%
2.2	Froyo	8	17.3%
2.3 - 2.3.2	Gingerbread	9	0.4%
2.3.3 - 2.3.7	Gingerbread	10	63.6%
3.1	Honeycomb	12	0.5%
3.2	Honeycomb	13	1.9%
4.0 - 4.0.2	Ice Cream Sandwich	14	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	10.7%

Tabela 2.1: Verzije mobilnega operacijskega sistema Android [4].

Imena verzij se določajo tako, da je začetna črka vsake nove verzije naslednja črka abecede. Nivo API (ang. application programming interface) ali programski vmesnik aplikacije [5] predstavlja številčno vrednost, ki enolično določa spremembo funkcij, ki jih ponuja verzija operacijskega sistema.

Android zagotavlja zbirko funkcionalnosti, ki aplikacijam omogočajo interakcijo z osnovnim sistemom Android. Okvir API sestavljajo:

- nabor paketov in razredov,
- nabor elementov XML in atributov za deklaracijo manifest datotek v kateri so definirane vse potrebne informacije (knjižnice, dovoljenja, komponente aplikacije ipd.) za delovanje aplikacije,
- nabor elementov XML in atributov za deklaracijo in dostop do virov,
- nabor dovoljenj, ki jih lahko zahtevajo aplikacije.

Android aplikacije so naprej kompatibilne. To pomeni, da aplikacije ustvarjene na starejših verzijah, delujejo tudi na novih verzijah. Aplikacije niso kompatibilne tudi za nazaj. Če smo določeno aplikacijo ustvarili za novo

verzijo, ni nujno da bo pravilno delovala na starejših verzijah operacijskega sistema Android.

Operacijski sistem Android vsebuje osnovne funkcije in lastnosti. Opišimo nekatere:

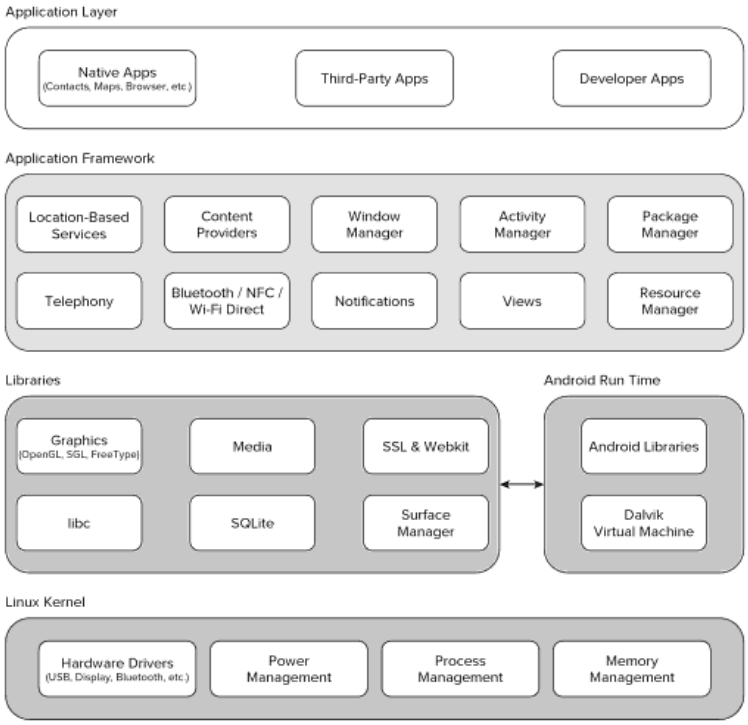
- postavitev zaslona - prilagodljiva je za VGA, 2D in 3D knjižnice, ki temeljijo na OpenGL ES 2.0 specifikaciji, ter delujejo standardnih zaslonih pametnih telefonov;
- shranjevanje - za shranjevanje podatkov se uporablja relacijska podatkovna baza SQLite;
- povezljivost - podpira tehnologije kot so GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC in WiMAX;
- sporočanje - podpira pošiljanje SMS in MMS, kot tudi naprednejši način pošiljanja sporočil preko strežnika Android Cloud To Device Messaging (C2DM);
- večjezičnost;
- spletni brskalnik - temelji na odprtokodnem ogrodju WebKit;
- podpora javi - čeprav je večina aplikacij napisanih v javi, na Androidu ni nameščen Java Virtual Machine (JVM), prav tako se ne izvaja prevedena bitna koda. Javanski razredi so prevedeni v izvršljivo datoteko Dalvik, ki jo poganja Dalvik Virtual Machine (DVK);
- multimedijska podpora - podpira naslednje audio/video formate: WebM, H.263, H.264,MPEG-4SP, AMR , AMR-WB , AAC, HE-AAC, MP3, MIDI, Ogg Vorbis, FLAC, WAV, JPEG, PNG, GIF, BMP, WebP;
- podpora za dodatno strojno opremo - možnost priklopa in delovanja različne strojne opreme kot so npr.: zaslon na dotik, GPS naprave, giroskopi, magnetometri, senzorji pritiska;
- večopravilnost - naenkrat lahko teče več aplikacij;

- večdotičnost - zaslon na dotik, ki zazna več točk dotika naenkrat;
- glasovno upravljanje - z glasovnimi ukazi lahko iščemo, kličemo, pišemo, itd.

2.2 Zgradba sistema Android

Mobilni operacijski sistem [6] je sestavljen iz naslednjih elementov (slika 2.1):

- jedro Linux - jedro zagotavlja abstrakcijo med strojno opremo naprave in ostalim delom operacijskega sistema;
- knjižnice - knjižnice so napisane v programskih jezikih C/C++ in tečejo na vrhu jedra;
- izvajalno okolje Android - vključuje knjižnice in Dalvik VM ter tako skrbi za izvajanje aplikacije na napravi;
- aplikacijsko ogrodje - zagotavlja dostop do strojne opreme in razredov, ki so potrebni za kreiranje aplikacije. Upravlja z uporabniškim vmesnikom in z aplikacijskimi viri;
- aplikacijski nivo - vse aplikacije se nahajajo na aplikacijskem nivoju in so zgrajene z istimi knjižnicami API.



Slika 2.1: Zgradba mobilnega operacijskega sistema Android.

2.3 Zgradba aplikacije Android

Android struktura [6] spodbuja ponovno uporabo komponent in omogoča objavo in izmenjavo aktivnosti, storitev in podatkov z ostalimi aplikacijami. Vsaka Android aplikacija vsebuje naslednje aplikacijske storitve:

- upravitelj aktivnosti - nadzira življenjski cikel aktivnosti;
- pogledi - uporabljajo se za izgradnjo uporabniških vmesnikov posamezne aktivnosti;
- upravitelj obveščanja - zagotavlja konsistenten mehanizem za obveščanje uporabnika;
- ponudnik vsebin - omogoča izmenjavo podatkov med aplikacijami;
- upravitelj virov - zagotavlja dostop do virov, kot so nizi in slike;

- namera - zagotavlja mehanizem, ki skrbi za prenos podatkov med aplikacijami in med njihovimi komponentami.

Vsaka Android aplikacija vsebuje naslednje komponente [7]:

- aktivnosti - aktivnost predstavlja en sam zaslon z uporabniškim vmesnikom. Primer: eno aktivnost imamo za pregled elektronskih sporočil, drugo aktivnost za pisanje sporočil in tretjo za branje elektronskih sporočil. Čeprav aktivnosti delujejo tako, da tvorijo povezano uporabniško izkušnjo, so med seboj neodvisne;

- storitve - so komponente, ki tečejo v ozadju in izvajajo dolgotrajne procese ali oddaljene procese. Storitve nimajo uporabniškega vmesnika;

- ponudniki vsebine - upravljajo z rabo podatkov aplikacije. Podatke shranimo v datotečni sistem, podatkovno bazo SQLite ali na internet. Preko ponudnikov vsebine aplikacije dostopajo ali urejajo podatke;

- sprejemniki dogodkov - je komponenta, ki se odziva na ravni celotnega sistema ob napovedi nekega dogodka. Večina dogodkov izvira iz sistema, kot je opozorilo o prazni bateriji, shranjeni sliki ipd. Nekatere dogodke pa sprožijo tudi aplikacije, npr. obvestilo o spremembi delnic v aplikaciji, ki prikazuje stanje na borzi ipd.

2.4 Razvoj aplikacije Android v Eclipse IDE razvojnem okolju

Eclipse IDE [8] je večjezično razvojno okolje za razvoj programske opreme. Razvojno okolje je večinoma napisano v programskem jeziku java in je sestavljeno iz integriranega razvojnega okolja in sistema razširitvenih vtičnikov. Omogoča razvoj programske opreme v programskem jeziku java, z nameščenimi vtičniki pa tudi v ostalih programskih jezikih kot so Ada, C, C++, COBOL, Haskell, Perl, PHP, Python, R, Ruby, Scala, Clojure, Groovy in Scheme. Uporablja se tudi za razvoj paketov programa Mathematica. Razvojno okolje vključuje tudi ostala javanska razvojna orodja, kot so JDT za javo, Eclipse CDT za C/C++, Eclipse PDT za PHP ipd. Z Eclipse IDE lahko razvijamo tudi aplikacije za Android, tako da namestimo Android SDK. Android SDK je paket, ki nam omogoča razvijanje, razhroščevanje in testiranje aplikacije Android. Paket vsebuje vsa potrebna orodja za razvoj aplikacije Android. Android SDK vsebuje:

- zbirko API,
- razvojna orodja,
- dokumentacijo,
- primere kode in
- spletno podporo.

Poglavje 3

Optimizacija z rojem delcev

Optimizacija z rojem delcev (ang. Particle Swarm Optimization) [9] je sprva ponazarjala simulacijo vedenja ptičje jate. PSO sta predstavila Kennedy in Eberhart [10], ki sta opisala številne filozofske vidike optimizacije z rojem delcev in inteligence rojev na splošno. PSO izvira iz dveh konceptov. Najprej potencialne rešitve sestavljajo delci, ki se premikajo skozi problemski prostor tako, da posamezni delci sledijo najbolj optimalnemu delcu. Delec sledi lokacijam v prostoru, ki so določene z najboljšo rešitvijo. Drugi idejni temelj je genetsko in evolucijsko programiranje. Od ostalih evolucijskih metod se razlikuje po tem, da ne vsebuje evolucijskih operatorjev, kot sta mutacija in križanje. S PSO lahko učinkovito rešujemo mnogo problemov. V PSO se delci gibljejo v preiskovalnem prostoru. Delci spreminjajo svoje pozicije posnemajoč uspešnejše sosednje delce. Vsak delec s svojim obnašanjem in iskanjem vpliva na ostale delce v roju, posledica takšnega obnašanja je, da se delci stohastično premikajo proti boljšim območjem v iskalnem prostoru. V optimizaciji z rojem delcev vsak delec predstavlja potencialno rešitev problema.

3.1 Optimizacija z rojem delcev v računalništvu

V računalništvu optimizacijo z rojem delcev predstavimo kot algoritem, ki uporablja pojme kot so: preiskovalni prostor, posamezni delci, roj delcev, parametri, določanje kakovosti delca itn. Delec je v preiskovalnem prostoru predstavljen z določeno hitrostjo in pozicijo, ki ju zapišemo v vektorski obliki.

Hitrost delca se izračuna po obrazcu:

$$v_{t+1}^{id} = v_t^{id} + C1 * (p_t^{id} - x_t^{id}) + C2 * (p_t^{ld} - x_t^{id}) + C3 * (p_t^{gd} - x_t^{id}), \quad (3.1)$$

kjer v_t^{id} predstavlja trenutno hitrost delca, p_t^{id} predstavlja pozicijo trenutno najboljše rešitve, p_t^{ld} predstavlja lokalno najboljšo rešitev, p_t^{gd} pa globalno najboljšo rešitev. Poziciji dodamo premike v smeri dobrih rešitev, ki jih dobimo s C1, C2 in C3 parametri.

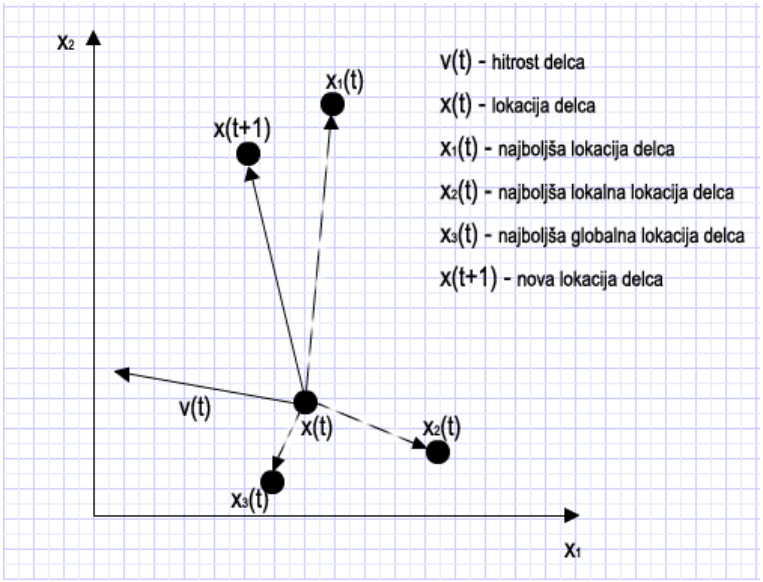
Pozicija delca se izračuna po obrazcu:

$$x_{t+1}^{id} = x_t^{id} + v_{t+1}^{id}, \quad (3.2)$$

kjer x_t^{id} predstavlja dosedanjo lokacijo delca, ki ji dodamo novo hitrost v_{t+1}^{id} in tako dobimo novo lokacijo delca x_{t+1}^{id}. Geometrijski prikaz spremembe pozicije delca je predstavljen na sliki 3.1.

Vsak delec pozna podatke o svoji dosedanji uspešnosti, zato hrani podatke o treh lokacijskih vektorjih [11]:

- svojo najboljšo lokacijo,
- lokalno najboljšo lokacijo oziroma najboljšo lokacijo svojih informatorjev in
- globalno najboljšo lokacijo.



Slika 3.1: Prikaz delca v času t in t+1.

Koraki, ki so potrebni v vsaki iteraciji optimizacije so naslednji:

1. inicializacija vseh delcev z naključno vrednostjo hitrosti in pozicije;
2. za vsak delec v roju se izračuna kvaliteta rešitve;
3. primerjava vseh delcev po kvaliteti rešitev in nastavljanje najboljših kvalitet in pozicij delcem;
4. iskanje najboljšega delca med svojimi informatorji in globalno najboljšega delca;
5. spreminjanje hitrosti in pozicije delcev po enačbah (3.1) in (3.2);
6. ponavljanje od 2. koraka naprej, dokler ustavitveni pogoj ni resničen.

Delovanje optimizacije z rojem delcev prikazuje naslednja psevdo koda:

```
function OSEBEKPSO(α, β, γ, δ, ϵ, velikostRoja)
    //α - utež vektorja hitrosti v
    //β - utež vektorja najboljše lastne vrednosti x*
    //γ - utež najboljše lokalne vrednosti x+
    //δ - utež najboljše globalne vrednosti x!
    //ϵ - korak premikanja delcev
    //velikostRoja - velikost roja

    P = []; //vektor delcev
    for i = 0 → velikostRoja do
        P_i = nov naključni delec;
        i ← i + 1;
    end for
    najboljši = null;
    repeat
        for i = 0 → velikostRoja do
            določiKvaliteto(P_i);
            if kvaliteta(P_i) > kvaliteta(najboljši) then
                najboljši = P_i ;
            end if
            i ← i + 1;
        end for
        for i = 0 → velikostRoja do
            x* = lokacija z dosedanjo največjo kvaliteto;
            x+ = lokacija z dosedanjo lokalno najboljšo kvaliteto;
            x! = lokacija z dosedanjo globalno najboljšo kvaliteto;
            for j = 0 → steviloDimenzij do
                b = random(β);
                c = random(γ);
                d = random(δ);
                v_{i,j} = α * v_{i,j} + b*(x*_j - x_{i,j}) + c*(x+_j - x_{i,j}) + d*(x!_j - x_{i,j});
                j ← j + 1;
            end for
            i ← i + 1;
        end for
        x_i = x_i + ϵ * v_i;
    until najboljši ni optimalen IN čas se ni iztekel
    return najboljši;
end function
```

Opis parametrov algoritma PSO:

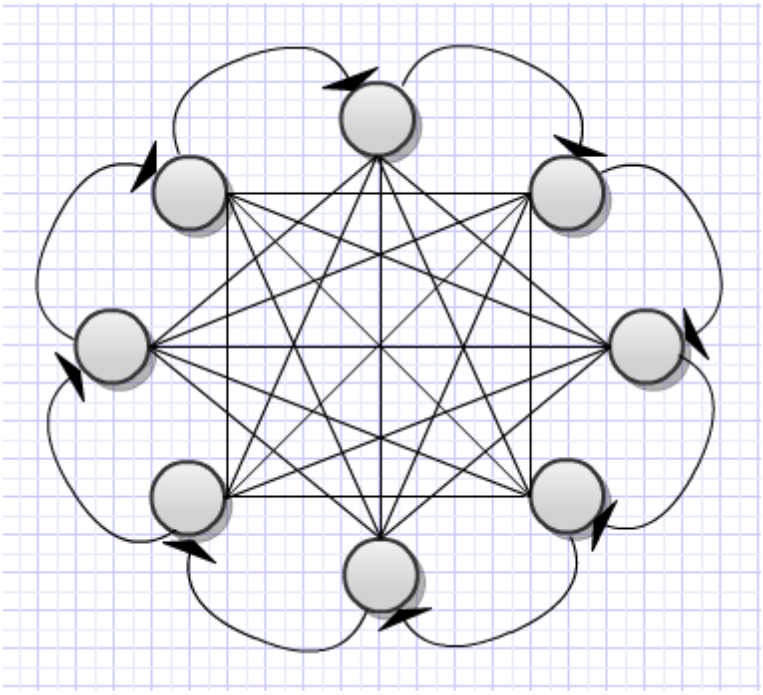
- α - delež vektorja hitrosti, ki ga upoštevamo pri računanju nove smeri vektorja;
- β - delež upoštevane najboljše lastne vrednost delca. Paziti moramo, da vrednost ni prevelika. Če je vrednost prevelika, dobimo požrešno individualno iskanje in tako ne dosežemo globalnega optimuma;
- γ - delež upoštevane najboljše vrednosti sosednjih delcev. Učinek deleža je med α in β. Če določimo več lokalnih skupin delcev, daje delež večjo težo globalnemu optimumu in manj lokalnemu;
- δ - delež upoštevane najboljše globalne vrednosti delcev. Če je vrednost prevelika, dobimo požrešno iskanje proti globalnemu optimumu in tako izpustimo možne boljše rešitve oziroma onemogočimo več ločenih iskanj. Običajno je ta vrednost postavljena na 0;
- ϵ - hitrost premikanja delcev. Pri prevelikih vrednostih prehitro napredujemo proti najboljšim lokacijam ali pa preskočimo dejansko najboljše. Pri manjših vrednostih lahko bolj natančno pridemo do optimuma a je iskanje počasnejše;
- velikostRoja - velikost roja delcev.

3.2 Strukture omrežij delcev

Delci v roju se učijo drug od drugega in se na podlagi tega premikajo tako, da so bolj podobni svojim boljšim sosedom [12]. Družbena struktura PSO je določena s prekrivanjem prostorov, kjer delci vplivajo drug na drugega. V resničnem svetu je to podobno obnašanju živali, kjer na posamezne živali vplivajo ostale živali v njihovi okolici in kjer imajo bolj uspešne večji vpliv.

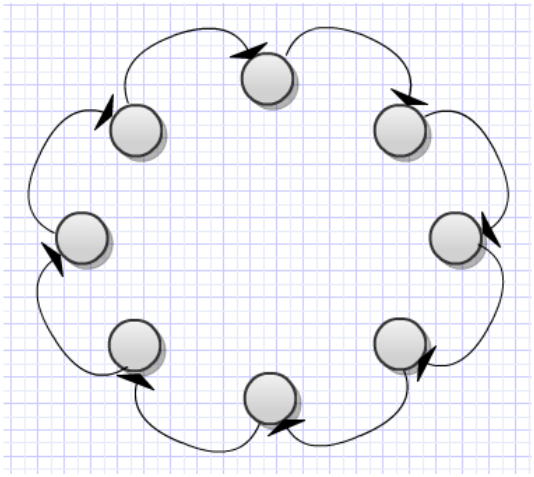
V PSO delci v isti okolici komunicirajo drug z drugim z izmenjavo informacij o uspešnosti oziroma kakovosti delcev v tej okolici. Na podlagi teh informacij se ti delci premaknejo v smer, za katero mislijo, da je najboljša. Pretok informacij preko družbenega omrežja odvise od stopnje povezanosti med posameznimi vozlišči v omrežju, stopnjo združenja in povprečno najmanjše razdalje med dvema vozliščema. V zelo povezanih socialnih omrežij lahko večina posameznikov komunicira med seboj. Posledica tega je hiter prehod informacije skozi družbeno omrežje, kar pomeni hitrejšo konvergenco kot pri manj povezanih omrežij. Potencialna slabost je slabo pokritje prostora oziroma optimizacija bolj verjetno pristane v lokalnem minimumu. Za PSO so bile razvite različne strukture omrežij, nekatere podrobneje opisujemo, saj so pomembne za implementacijo optimizacije, ostale pa le omenilno:

- zvezdna struktura - vsi delci so povezani, kot kaže slika 3.2. Vsak delec lahko komunicira s poljubnim drugim delcem in je lahko najboljša rešitev v celotnem roju. Prva implementacija PSO, ki je uporabila zvezdno družbeno strukturo, se imenuje gbest oziroma globalno najboljši PSO;



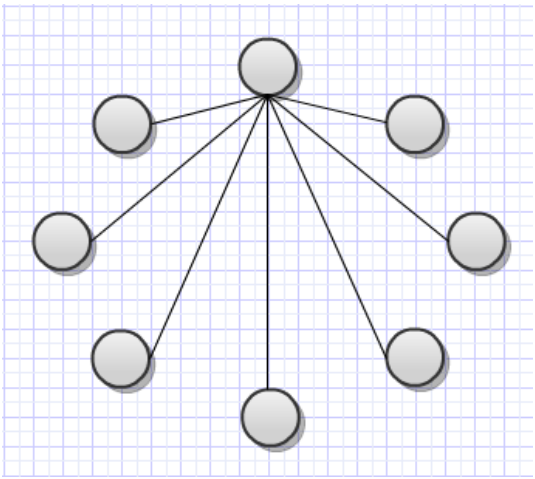
Slika 3.2: Zvezdna struktura.

- obročna struktura - vsak delec komunicira z nN neposrednimi sosedi. Ko je nN = 2, vsak delec komunicira natanko z dvema neposrednima sosedoma (slika 3.3). Vsak delec želi posnemati svojega najboljšega soseda tako, da se premakne k najboljši rešitvi v okolici. Na sliki opazimo, da se okolice prekrivajo, kar omogoča izmenjavo informacij med okolicami in na koncu konvergenco k eni rešitvi. Ker je prenos informacij skozi omrežje počasnejši je tudi konvergenca počasnejša, kar omogoči večjo pokritost iskalnega prostora, kot pri zvezdni strukturi. Obročna struktura zagotavlja boljše rezultate glede problemov, ki imajo več rešitev;



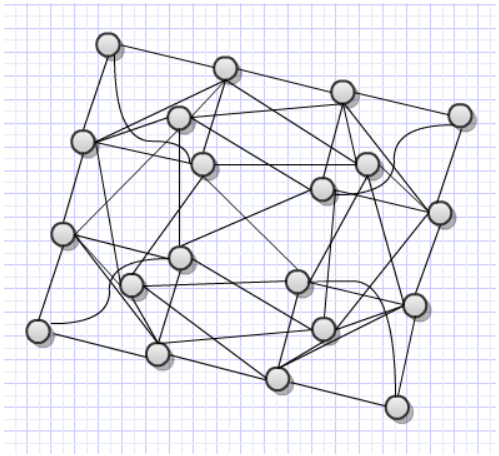
Slika 3.3: Obročna struktura.

- kolesna struktura - delci v okolici so ločeni med seboj. En delec služi kot kontaktni delec, skozi katerega se pretakajo vse informacije (slika 3.4). Kontaktni delec primerja zmožnosti delcev v okolici in prilagodi svoj položaj v smeri najboljšega delca. Če nova lokacija izboljša zmogljivost roja, sporoči to vsem delcem v okolici. Kolesno družbeno omrežje upočasni širjenje dobrih rešitev skozi roj;



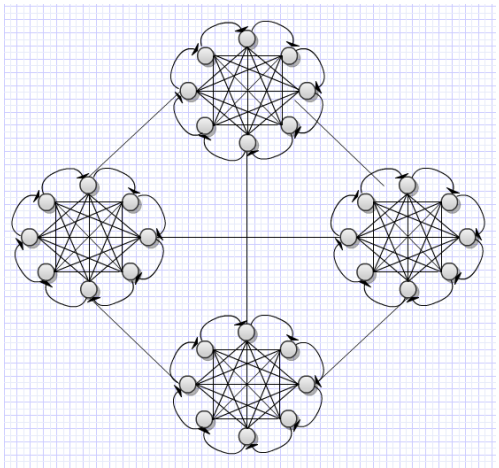
Slika 3.4: Kolesna struktura.

- piramidna struktura - ustvarja trodimenzionalni žični okvir, kot kaže slika 3.5;

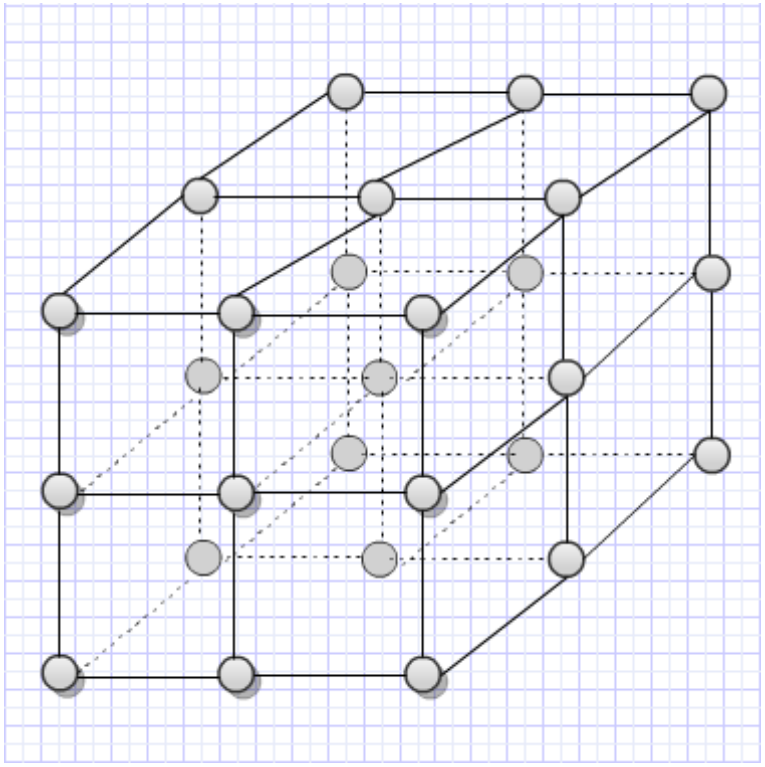


Slika 3.5: Piramidna struktura.

- struktura s štirimi kopicami - v strukturi so povezane štiri kopice (vsaka z vsako). Znotraj vsake kopice so vsi delci povezani med sabo (slika 3.6);



Slika 3.6: Struktura s štirimi kopicami.



Slika 3.7: Von Neumannova struktura.

- Von Neumannova struktura - vsi delci so povezani v mrežo strukture na sliki 3.7. Von Neumannova družbena struktura se je v empiričnih študijah izkazala za najboljšo med družbenimi strukturami.

Različne strukture so nastale zato, ker ne obstaja struktura, ki bi reševala vse probleme. Glede na problem določimo in izberemo družbeno strukturo, ki bo najbolje ustrezala problemu.

3.3 Globalno najboljši PSO

Pri globalno najboljšem PSO je za vsak delec soseska enaka kar celotnemu roju. Globalno najboljši PSO za topologijo uporablja zvezdno strukturo. PSO konvergira hitro in omogoča hiter pretok informacij skozi omrežje. Težava je, da se lahko včasih ujame v lokalnem optimumu in tako ne dobimo globalnega optimuma. Največkrat ga uporabimo pri problemih z eno možno rešitvijo. Algoritem globalno najboljšega PSO je predstavljen z naslednjo psevdo kodo:

Kreiranje in inicializacija roja delcev;
repeat
  for $i = 0 \rightarrow velikostRoja$ **do**
    // nastavljanje najboljše pozicije delca
    if $f(x_i) < f(y_i)$ **then**
      $y_i = x_i$;
    end if
    // nastavljanje globalno najboljše pozicije
    if $f(y_i) < f(y')$ **then**
      $y' = y_i$;
    end if
    $i \leftarrow i + 1$;
  end for
  for $i = 0 \rightarrow velikostRoja$ **do**
    $v_{ij}(t+1)$=v_{ij}(t)+$c_1 r_{1j}$(t)[y_{ij}(t)−x_{ij}(t)]+$c_2 r_{2j}$(t)[y'_j(t)−x_{ij}(t)];
    $x_i(t+1)$=$x_i(t)$+$v_i(t+1)$;
    $i \leftarrow i + 1$;
  end for
until ustavitveni pogoj ni resničen

3.4 Lokalno najboljši PSO

Lokalno najboljši PSO za topologijo uporablja obročno strukturo. Ker je prenos informacij skozi omrežje počasnejše je tudi konvergenca počasnejša, kar omogoči večjo pokritost iskalnega prostora, kot pri zvezdni strukturi oziroma globalno najboljšem PSO. Obročna struktura zagotavlja boljše rezultate v smislu kakovosti najdenih rešitev, pri problemih, ki imajo več rešitev. Algoritem lokalno najboljšega PSO je predstavljen z naslednjo psevdo kodo:

Kreiranje in inicializacija roja delcev;
repeat
  for $i = 0 \rightarrow velikostRoja$ **do**
    // nastavljanje najboljše pozicije delca
    if $f(x_i) < f(y_i)$ **then**
      $y_i = x_i$;
    end if
    // nastavljanje lokalno najboljše pozicije
    if $f(y_i) < f(y'_i)$ **then**
      $y'_i = y_i$;
    end if
    $i \leftarrow i + 1$;
  end for
  for $i = 0 \rightarrow velikostRoja$ **do**
    $v_{ij}(t+1) = v_{ij}(\mathrm{t}) + c_1 r_{1j}(\mathrm{t})[y_{ij}(\mathrm{t}) - x_{ij}(\mathrm{t})] + c_2 r_{2j}(\mathrm{t})[y'_{ij}(\mathrm{t}) - x_{ij}(\mathrm{t})]$;
    $x_i(t+1) = x_i(t) + v_i(t+1)$;
    $i \leftarrow i + 1$;
  end for
until ustavitveni pogoj ni resničen

Poglavje 4

Vizualizacija

V tem poglavju predstavimo idejo teme, motivacijo in glavni problem teme. Predstavimo zastavljene cilje in glede na njih tudi specifikacije za implementacijo. Opisujemo proces načrtovanja optimizacije in vizualizacije algoritma, postavljanje zahtev in specifikacij, izdelavo aplikacije in probleme, s katerimi smo se soočali pri izdelavi, ter testiranje izdelane aplikacije.

4.1 Ideja vizualizacije

Ideja vizualizacije algoritmov se je pojavila pri učenju različnih algoritmov. V večini primerov smo se delovanje algoritmov učili tako, da smo na papir zapisovali sled algoritma in ugotavljali, kako algoritem deluje. Želeli smo enostavnejši način, ki bi skrajšal čas učenja algoritma, zato smo se odločili za vizualizacijo algoritma, ki je uporabniku bolj prijazna. Zaradi razvoja pametnih telefonov in tablic smo se odločili za vizualizacijo algoritma na mobilni platformi. Tako smo želeli razširiti vizualizacijo algoritmov iz spletnih in namiznih aplikacij tudi na mobilne naprave in tablice. Želeli smo izdelati hiter, uporabniku prijazen dostop do vizualizacije algoritma in mu omogočiti lažje razumevanje delovanja algoritma. Drugi cilj je spodbuditi razvoj vizualizacij algoritmov na mobilnih platformah, saj menimo, da bo v prihodnosti več poudarka na izobraževanju preko prenosnih in mobilnih naprav. Za imple-

mentacijo vizualizacije algoritma na mobilni platformi smo se odločili tudi zaradi tega, ker smo želeli uporabnikom - študentom ponuditi nekaj novega pri učenju in razumevanju določene snovi. Optimizacijo z rojem delcev smo izbrali zato, ker je algoritem razmeroma nov in ker si ga lahko predstavljamo tudi v naravi, kar omogoča lažje razumevanje problema. Problem algoritma je iskanje najbolj optimalne točke v nekem prostoru. Problem lahko ponazorimo z vedenjem jate rib ali ptic v naravi. V določenem prostoru se giblje jata, ki sodeluje med sabo in išče hrano. Ko osebek iz jate najde hrano, o tem obvesti bližnje sosede in tako se med osebki prenese informacija, kje se nahaja hrana oziroma kje je najboljša točka v prostoru iskanja.

4.2 Analiza zahtev

Zadali smo si nekatere zahteve in funkcionalnosti, ki naj bi jih vizualizacija podpirala. Vizualizacijo smo želeli implementirati na mobilni platformi Android, z najbolj razširjeno verzijo. Želeli smo upoštevati vse omejitve mobilnih naprav, tako fizičnih kot funkcionalnih. Aplikacija je namenjena uporabnikom, ki želijo spoznati algoritem optimizacije z rojem in si bolje predstavljati delovanje algoritma. Za uporabo aplikacije registracija ni potrebna.

Delovanje aplikacije naj ne bo odvisno od povezave z internetom, ker zato ni potrebe in ker lahko pride do nepotrebnih stroškov, predvsem pri prenosu podatkov v tujini. Aplikacija naj vsebuje opis parametrov algoritma, ki jih lahko nastavimo. Aplikacija naj omogoča dva načina vizualizacije delovanja optimizacije. V prvem načinu naj prikaže delovanje optimizacije dvodimenzionalnem prostoru, kot animacijo korakov izvajanja algoritma. V drugem načinu naj omogoča prikaz spremembe posameznega koraka določenega delca z vsemi vektorji. Optimizacija z rojem delcev vsebuje različne nastavljive parametre, npr. število ponovitev korakov optimizacije, določanje velikosti populacije, določanje pospešitvenih koeficientov, določanje ohranitve trenutnega vektorja, določanje hitrosti optimizacije in izbiro funkcije računanja

kakovosti, ki se uporabljajo pri optimizaciji. Vizualizacija naj se izvede v sprejemljivem času in naj ne prihaja do zakasnitev pri izvedbi optimizacije in animacije. Velikost parametrov naj ne bo omejena in naj ne vpliva na hitrost delovanja aplikacije. Uporabniški vmesnik naj bo intuitiven in uporabniku prijazen. Izbrana orodja za razvoj naj omogočajo pisanje v programskem jeziku java, napisana koda naj podpira večuporabnost in naj bo napisana v skladu s pravili kodiranja. Uporabniški vmesnik aplikacije naj bo ločen z objektnim modelom. Mobilna platforma naj bo operacijski sistem Android, aplikacija pa mora vsebovati elemente, ki se uporabljajo v operacijskem sistemu Android. Privzeto stanje parametrov algoritma naj uporabniku takoj prikaže značilnosti algoritma. Aplikacija mora vsebovati kratka navodila, opis delovanja, razlago parametrov in primerjavo različnih stanj optimizacije.

4.3 Analiza specifikacije vizualizacije optimizacije z rojem delcev

Glede na naše zahteve smo določili:

- Za mobilni operacijski sistem smo si izbrali Android verzije 2.3, ki ga ima trenutno največ uporabnikov Androida. Aplikacija bo delovala tudi na vseh novejših verzijah mobilnega operacijskega sistema Android, ker je kompatibilen navzgor (glej tabelo 2.1);

- Fizične omejitve mobilne naprave so minimalne, priporočljiva je uporaba vsaj 4.3 palčnega zaslona, kar je dovolj za optimalen prikaz vizualizacije algoritma;

- Za uporabo aplikacije ni potrebna predhodna registracija, ker jo hkrati lahko uporablja samo ena oseba na mobilni napravi. Za delovanje aplikacije ni potreben prenos podatkov oziroma povezava z internetom, ker trenutno ni implementirana povezava z oddaljeno bazo, do katere bi več

uporabnikov dostopalo s svojim uporabniškim imenom. Uporabniki bi na primer primerjali najboljše rešitve ali nabor različnih vrednosti parametrov;

- Aplikacija vsebuje intuitiven uporabniški vmesnik, ki je sestavljen iz glavnega menija, ki za vsako izbiro odpre novo aktivnost na mobilni napravi, na primer zagon animacije, pregled nastavitev, opis parametrov, opis delovanja aplikacije in kratek opis algoritma;

- Aplikacija omogoča dve vizualizaciji optimizacije roja delcev; v prvi aplikacija prikaže animacijo celotnega procesa iskanja optimalne točke v dvodimenzionalnem prostoru. V drugem načinu aplikacija omogoča animacijo poljubnega delca, tako da prikaže korake tega delca, od njegove začetne pozicije do končne pozicije, ki je približek najboljše pozicije. Pri vsakemu delcu so prikazani vektorji, od katerih je odvisna naslednja pozicija delca. Pri posameznih delcih vidimo tudi legendo, ki pove, katere vektorje smo upoštevali pri računanju naslednje pozicije. Vektorji se ločeni z različnimi barvami;

- Aplikacija omogoča nastavitev različnih parametrov, ki so potrebni za delovanje optimizacije roja delcev. V aplikaciji se lahko nastavlja velikost populacije v prostoru, ki nam pove število aktivnih delcev, nastavimo lahko število ponovitev korakov optimizacije;

- Vizualizacija se večinoma izvede v sprejemljivem času in ne prihaja do pretiranih zakasnitev razen v primerih, kot smo jih navedli zgoraj. Vrednosti parametrov niso omejene, vendar je potrebno paziti, da vnesemo primerne vrednosti, sicer ne dobimo smiselne vizualizacije;

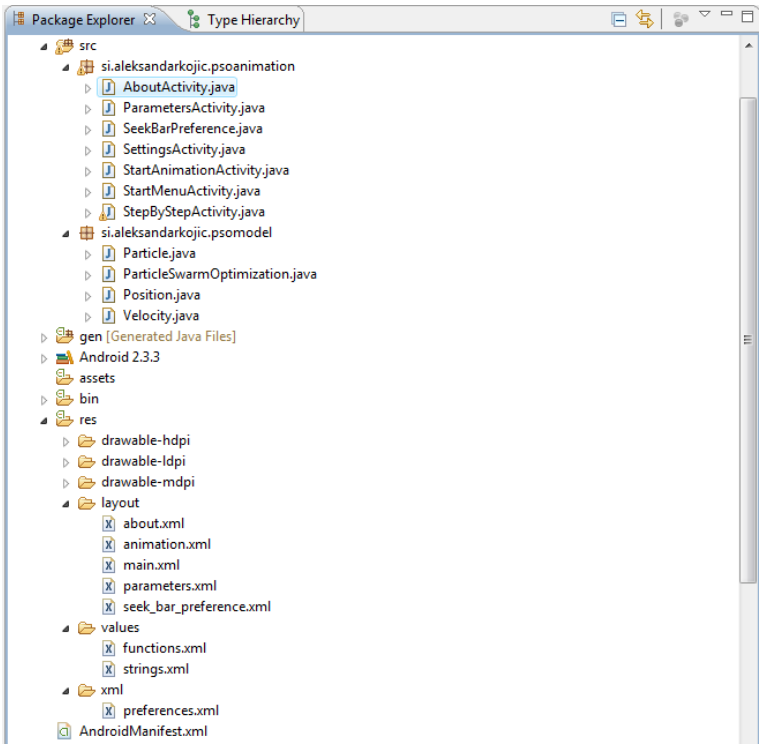
- Aplikacija je napisana v programskem jeziku java v orodju Eclipse IDE s pomočjo Andorid SDK, ki vsebuje različne knjižnice in funkcije, s katerimi si pomagamo pri razvoju aplikacij v Andoridu. Koda podpira večuporabnost, kar pomeni, da se večkrat uporabljena koda ne podvaja. Koda temelji na objektnem modelu, kjer so lastnosti in metode

potrebne za delovanje algoritma, združene v objektih. Uporabniški vmesnik je ločen od objektnega modela, saj je tako lažje vzdrževati kodo in odpravljati napake;

- Začetno stanje parametrov je nastavljeno na privzete vrednosti, kar pri zagonu aplikacije daje dober pregled delovanja optimizacije;
- Aplikacija ponuja kratek opis delovanja optimizacije z rojem delcev;
- Aplikacija omogoča kratko pomoč, kjer so opisani parametri in njihov vpliv na optimizacijo.

4.4 Načrtovanje in izdelava aplikacije

Ko smo načrtovali aplikacijo, smo želeli objektni model ločiti od uporabniškega vmesnika, kar je na Androidu enostavno mogoče. Na eni strani imamo poslovno logiko, ki jo predstavljajo javanski razredi, na drugi pa uporabniški vmesnik, ki je predstavljen z XML datotekami. Vmes so še javanski razredi, ki predstavljajo aktivnosti in skrbijo za prikaz uporabniškega vmesnika aplikacije in uporabo objektnega modela aplikacije. Eclipse IDE z vtičnikom Android SDK nam omogoča uporabo že napisanih funkcij, tako da to olajša programiranje aplikacije. Odločili smo se, da ne bomo uporabili podatkovnih zbirk, saj to ni potrebno. V primeru, da bi želeli razširiti aplikacijo in ohraniti hitrost aplikacije, bi za namišljeno bazo pri majhni količini podatkov uporabili kar generirane XML datoteke, ki bi predstavljale zapise v bazi, lahko pa bi uporabili bazo SQLite, ki je prisotna na mobilni napravi. Seznam paketov, razredov in ostalih datotek, ki smo jih uporabili pri izdelavi aplikacije vidimo na sliki 4.1.



Slika 4.1: Seznam paketov, razredov in ostalih datotek uporabljenih pri aplikaciji.

Vse razrede, aktivnosti, pakete in ostale pripadajoče datoteke predstavimo v nadaljevanju.

Paket si.aleksandarkojic.psomodel vsebuje razrede, ki predstavljajo objektni model aplikacije. Razredi so strukturirani tako, da so pomensko podobne in povezane funkcije, spremenljivke ipd. v enem razredu. Paket vsebuje naslednje objekte:

- **Particle.java** - predstavlja delec iz roja. Vsak delec nosi informacijo o poziciji, najboljši lastni poziciji, lokalno najboljši poziciji in o trenutno najboljšem delcu. Vsak delec vsebuje informacijo o trenutni hitrosti in najboljši hitrosti. Za delovanje optimizacije je pomembna funkcija kvalitete, zato vsak delec vsebuje informacije o trenutni kvaliteti delca, o najboljši kvaliteti delca in o lokalno najboljši kvaliteti delca. Objekt

lahko inicializiramo z obstoječim delcem in mu spreminjamo vrednosti, na primer pozicije, hitrosti in kvalitete delca. Pri računanju kvalitete delca je ta odvisna pozicije delca. Znanih je več funkcij vrednotenja. Uporabili smo naslednje znane funkcije vrednotenja: Peaksova funckija, sinusna funkcija;

- **Position.java** - razred predstavlja lokacijo delca v prostoru. Ker optimizacija temelji na evklidski razdalji oziroma prikazujemo delce v 2D prostoru imamo x in y koordinati predstavljeni kot par vrednosti. Razred omogoča nastavljanje vrednosti x in y;

- **Velocity.java** - predstavlja hitrost delca. Podobno kot lokacijo, hitrost predstavlja par vrednosti x in y. Hitrost delca je v animaciji predstavljena kot vektor, ki je enak razliki med vektorjema razdalje in hitrosti delca;

- **ParticleSwarmOptimization.java** - razred predstavlja samo optimizacijo z rojem delcev. Vsebuje lastnosti, kot so velikost populacije, število ponovitev korakov, parametre algoritma in ostale potrebne parametre. Objekt inicializiramo tako, da mu podamo potrebne parametre za izvedbo optimizacije. Vsebuje metodo za inicializacijo roja delcev oziroma generiranje naključnih začetnih vrednosti delcev, metodo za izvedbo optimizacije z vsemi pripadajočimi parametri in lastnostmi in metodo, ki nam omogoča spremljanje sprememb enega določenega delca, ki ga bomo uporablili pri animiciji.

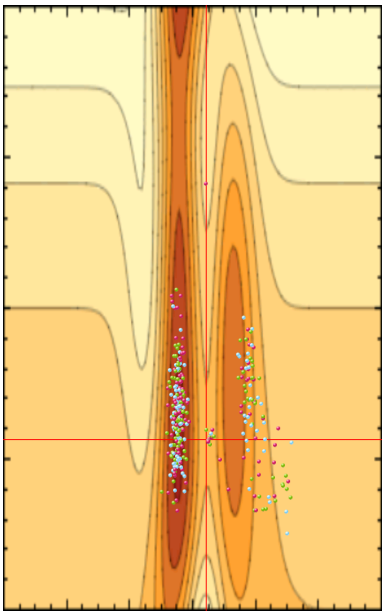
Paket si.aleksandarkojic.psoanimation vsebuje razrede, ki predstavljajo aktivnosti oziroma ostale pomožne razrede, ki so potrebni pri vizualizaciji algoritma. Vsaka aktivnost predstavlja eno možnost prikaza v aplikaciji, zato ima vsaka aktivnost svoj uporabniški vmesnik. Paket vsebuje naslednje razrede, ki predstavljajo aktivnosti:

- **StartMenuActivity.java** - razred predstavlja začetno aktivnost, ki se prikaže ob zagonu aplikacije. Aktivnost vsebuje začetni logotip in meni (slika 4.2), preko katerega lahko dostopamo do ostalih aktivnosti v aplikaciji;

Slika 4.2: Začetna aktivnost aplikacije.

- **StartAnimationActivity.java** - razred predstavlja aktivnost, ki prikaže simulacijo delovanja optimizacije z rojem delcev (slika 4.3). V ozadju vidimo območje, po katerem se delci gibajo in iščejo globalni optimum. Delci istih barv pripadajo lokalnim skupinam, ki si delijo informacije o najboljših lokacijah delcev v skupini. S tem preprečimo prehitro konvergenco k globalnem optimumu in se lahko izognemo rešitvi, ki obtiči

v lokalnem optimumu;



Slika 4.3: Aktivnost aplikacije, ki prikazuje animacijo algoritma.

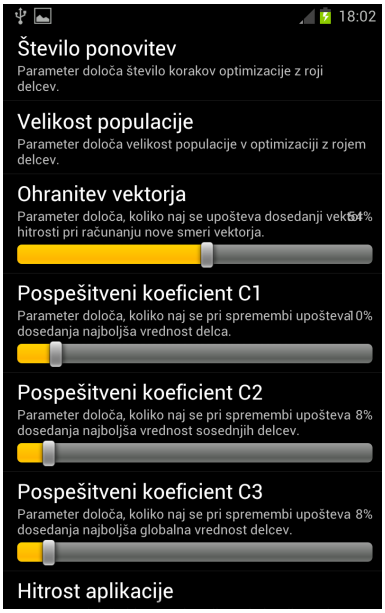
Za prikaz vsebine aktivnost uporablja interni razred Animation, ki razširja vgrajeni razred SurfaceView, ki se v Androidu uporablja za risanje po površinah. Spreminjamo lahko obliko površine in velikost. Razred je uporaben za animacije, ker omogoča direktno posodobitev površine. Kar želimo prikazati vstavimo med dvema klicema surfaceHolder.lockCanvas() in surfaceHolder.unlockCanvas(). Primer uporabe razreda SurfaceViewa je na sliki 4.4;

```
public class RenderingView extends SurfaceView implements Runnable {
        Thread thread = null;
        SurfaceHolder holder;
        volatile boolean running = false;

        public RenderingView(Context context) {
                super(context);
                holder = getHolder();
        }

        public void resume() {
                running = true;
                thread = new Thread(this);
                thread.start();
        }

        public void run() {
                while (running) {
                        if (!holder.getSurface().isValid())
                                continue;

                        Canvas canvas = holder.lockCanvas();
                        // Use canvas to draw things here
                        holder.unlockCanvasAndPost(canvas);
                }
        }

        public void pause() {
                running = false;
                while (true) {
                        try {
                                thread.join();
                        } catch (InterruptedException e) {
                                // wait
                        }
                }
        }
}
```

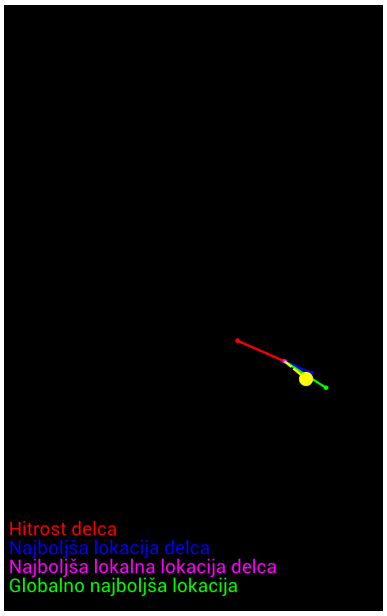
Slika 4.4: Primer uporabe SurfaceViewa.

- **SettingsActivity.java** - razred oziroma aktivnost predstavlja in prikazuje nastavitve parametrov, ki so potrebni za optimizacijo z rojem delcev. Aktivnost predstavljajo različni elementi, s katerimi nastavimo parametre optimizacije (slika 4.5); Razred SettingsActivity.java je implementiran tako, da razširja vgrajeni razred PreferenceActivity. Razred PreferenceActivity omogoča shranjevanje nastavljenih parametrov, ne da bi bila potrebna povezava do baze ali diska. Vse spremembe nastavitev parametrov se samodejno shranjujejo v SharedPreferences, do katerih dostopamo s klicem getDefaultSharedPreferences(). Hierarhijo nastavitev smo oblikovali tako, da smo ustvarili XML datoteko, ki vsebuje vrednosti parametrov. Nastavitve iz XML datoteke smo dodali s klicem addPreferenesFromResource(int);

- **StepByStepActivity.java** - aktivnost predstavlja animacijo naključno določenega deleca iz optimizacije. Pri vsakem koraku prikažemom, kje se nahaja delec, njegov vektor hitrosti, najboljšo lokacijo delca, najboljšo globalno lokacijo, najboljšo lokalno lokacijo in novo lokacijo delca. Tako uporabnik lažje razume delovanja optimizacije, saj lahko vidi, kako se premikajo delci in kako se računajo njihove nove lokacije (slika 4.6). Aktivnost vsebuje legendo, ki pove, kateri elementi animacije so določene barve. Za animacijo korakov smo uporabili razširjen vgrajeni razred SurfaceView, ki smo ga opisali zgoraj;

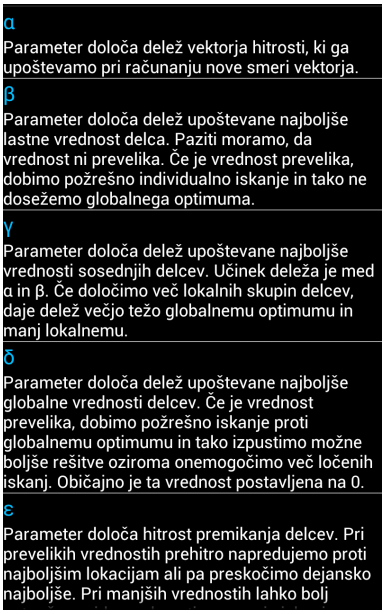


Slika 4.5: Aktivnost, ki prikazuje nastavitve parametrov.

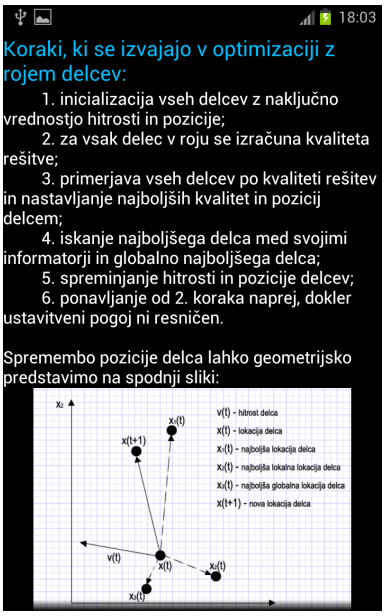


Slika 4.6: Aktivnost, ki prikazuje animacijo posameznega koraka algoritma.

- **ParametersActivity.java** - aktivnost vsebuje zbrane opise parametrov, ki so potrebni pri optimizaciji z rojem delcev (slika 4.7);
- **AboutActivity.java** - aktivnost vsebuje kratek opis algoritma in kratka navodila, kako se aplikacija uporablja. Vsebuje tudi opis optimizacije za lažjo predstavo o delovanju aplikacije (slika 4.8);



Slika 4.7: Aktivnost, ki prikazuje opis parametrov.



Slika 4.8: Aktivnost, ki prikazuje opis in navodila algoritma aplikacije.

- **SeekBarPreference.java** - za nekaj parametrov smo menili, da bi jih najlažje spreminjali z drsnikom. Takšna kontrola za nastavitve ne obstaja, zato jo je potrebno ročno implementirati. Uporabljena je pri parametrih za hitrost aplikacije, pospešitvene kooeficiente ipd. Razširjali smo vgrajeni razred Preference in implementirali metode OnSeekBarChangeListener, ki skrbijo za nastavljanje spremenjenih vrednosti drsnika in shranjujejo vrednost drsnika.

Uporabniški vmesniki aplikacije so predstavljeni kot XML datoteke v projektu, kjer se nahaja aplikacija. Pomembno vlogo pri aplikacijah v Androidu igra označevalni jezik XML. XML je preprost računalniški jezik, ki se uporablja za opisovanje strukturiranih podatkov [13]. Razdeljen je na tri dele:

- **podatkovni** - vanj shranimo podatke z željenimi oznakami oziroma značkami;
- **deklarativni** - skrbi za to, da lahko pri dodajanju novih podatkov vidimo, kaj značke predstavljajo;
- **predstavitveni** - z njim oblikujemo izpis podatkov.

V Androidu XML datoteke uporabljamo za več stvari. Lahko jih generiramo za shranjujevanje podatkov in jih uporabimo kot podatkovno bazo aplikacije, lahko jih uporabimo za shranjevanje virov aplikcije, kot so nizi in tabele, največkrat pa jih uporabljamo za postavitve (ang. layout) aktivnosti. Postavitev v aplikacijah Android predstavlja uporabniški vmesnik z vsemi pripadajočimi elementi oziroma gradniki. V postavitvi definiramo, kako so elementi prikazani uporabniku. V XML lahko ustvarimo uporabniški vmesnik z vsemi elementi, ki bodo prikazani uporabniku. Vsaka postavitvena datoteka (Slika 4.9) vsebuje korenski element, ki je objekt razreda View oziroma ViewGroup. Ko definiramo korenski element, lahko dodajamo poljubne ostale elemente in gradnike, ki predstavljajo uporabniški vmesnik aktivnosti.

Poznamo več postavitvenih gradnikov:

- **LinearLayout** - vsi elementi so poravnani v isti smeri, bodisi vertikalno ali horizontalno.
- **RelativeLayout** - elementom določimo poljubne relativne pozicije.
- **ListView** - prikazuje elemente kot pomični seznam.
- **GridView** - prikazuje elemente v dvodimenzionalni pomični mreži.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Slika 4.9: Primer uporabe postavitve v Androidu.

V naši aplikaciji uporabniški vmesniki predstavljajo naslednje datoteke:

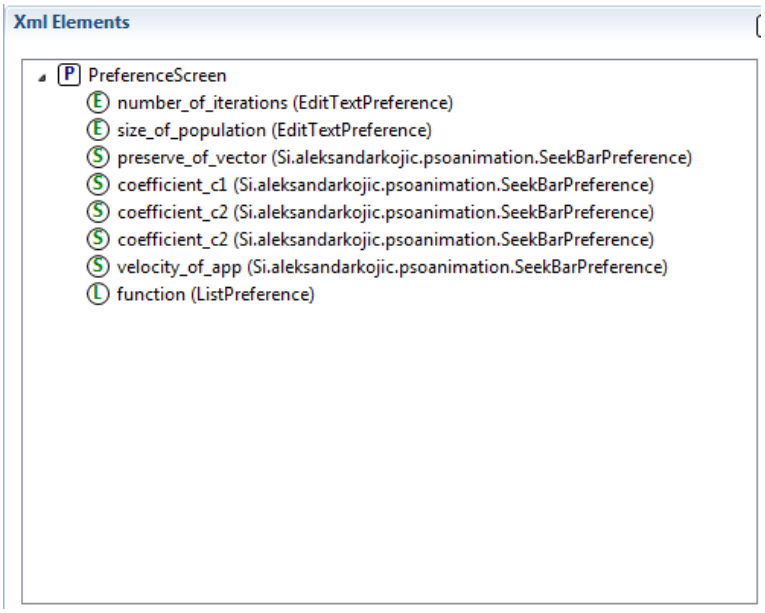
- **main.xml** - vsebuje uporabniški vmesnik za začetno aktivnost aplikacije. Vsebuje logotip z imenom aplikacije in meni, ki je sestavljen iz gumbov. Vsak gumb odpre naslednjo aktivnost, ki ima svoj uporabniški vmesnik. Zgradba postavitve v XML datoteki je na sliki 4.10;

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:weightSum="7">

    <ImageView
        android:id="@+id/imageViewPSOLogo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:src="@drawable/pso_logo1"
        android:contentDescription="@string/app_name"/>

    <LinearLayout
        android:id="@+id/LinearLayoutMainView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_weight="5"
        android:baselineAligned="false"
        android:orientation="vertical"
        android:weightSum="5">
```

Slika 4.10: Zgradba postavitve v main.xml datoteki.

- **animation.xml** - vsebuje uporabniški vmesnik za animacijo optimizacije z rojem delcev. Sestavljena je samo iz postavitvenega elementa, ker za prikaz ostalih elementov, ki so vidni uporabniku, poskrbimo v kodi razreda, ki predstavlja aktivnost;
- **parameters.xml** - uporabniški vmesnik za aktivnost, kjer prikazujemo opise posameznih parametrov;
- **about.xml** - prikazuje kratek opis algoritma in kratka navodila, kako se uporablja aplikacija;

- **seek_bar_preference.xml** - definira nastavitev, ki vsebuje drsnik, ki ga uporabimo pri definiranju uporabniškega vmesnika za nastavitve parametrov;
- **preferences.xml** - uporabniški vmesnik nastavitve parametrov. Korenski element je PreferenceScreen, kar predstavlja aktivnost z nastavitvami. Za predstavitev parametrov so uporabljeni izbirni seznam, drsnik, izbira in vnosna številčna polja. Nastavitve so predstavljene po modelu s slike 4.11;



Slika 4.11: Zgradba xml datoteke, ki predstavlja nastavitve parametrov.

- **functions.xml** - predstavlja vire aplikacije, ko jih uporabimo za izbiro ustrezne funkcije kakovosti. Sestavljena je iz tabele nizov, kjer ima vsak niz točno določeno vrednost. Zgradba XML datoteke je na sliki 4.12.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array
        name="list_functions">
        <item>Step Function</item>
        <item>Peaks Function</item>
        <item>Sinc Function</item>
        <item>Sphere Function</item>
    </string-array>

    <string-array
        name="list_values">
        <item>1</item>
        <item>2</item>
        <item>4</item>
        <item>5</item>
    </string-array>
</resources>
```

Slika 4.12: Zgradba xml datoteke, ki predstavlja vire aplikacije.

4.5 Testiranje aplikacije

Testiranje aplikacije smo začeli načrtovati hkrati z načrtovanjem aplikacije. Pripravili smo testne scenarije glede na funkcionalnosti aplikacije in glede na predvideno obnašanje uporabnikov. Želeli smo testirati vsak načrtovan del implementacije z različnimi mobilnimi napravami z več različnimi verzijami sistema Android. Pri novih funkcionalnostih smo aplikacijo testirali na emulatorju, ki ga ponuja orodje Eclipse IDE in na mobilni napravi. Pri testiranju smo zaznali nekaj napak, ki so bile posledica napak pri programiranju, pri načrtovanju in izdelavi uporabniških vmesnikov aplikacije. Aplikacijo smo testirali na mobilnih napravah Samsung I9100 Galaxy S II z resolucijo 480 x 800 in na operacijskem sistemu Android 2.3.6 na mobilni napravi Samsung Galaxy Note N7000, katerega resolucija je 800 x 1280 in operacijski sistem Android 2.3.6 oziroma Android 4.0.3.

Poglavje 5

Sklepne ugotovitve

V diplomski nalogi smo predstavili mobilni operacijski sistem Android in njegovo uporabo. Predstavljen je bil razvoj aplikacije za Android in in potrebna orodja. Predstavili smo glavne značnilnosti optimizacije z rojem delcev, njeno uporabo v naravi in v računalništvu. Opisali smo grobo delovanje optimizacije in to prikazali s psevdo kodo. Opisali smo dva najbolj uporabljena pristopa pri implementaciji optimizacije z rojem delcev in sicer globalni in lokalni PSO.

Glede na specifikacije in zahteve smo načrtovali in izdelali aplikacijo, ki prikazuje delovanje optimizacije z roji delcev. V prihodnosti bi lahko razširili aplikacijo, da bi se uporabljala tudi na spletni strani ali kot namizna aplikacija na računalniku. Dodali bi lahko tudi še vizualizacijo podobnih algoritmov kot je kolonija mravelj in ostalih algoritmov, ki spadajo v inteligenco rojev. Zbrane algoritme bi primerjali glede časa in kakovosti najboljše rešitve. V primerih, ko gre za veliko populacijo delcev, prihaja do manjših zakasnitev pri izvedbi optimizacije; to bi lahko izboljšali z optimizacijo napisane kode.

Z delovanjem aplikacije smo zadovoljni, trenutno jo uporablja še nekaj oseb, ki so nam pomagale pri testiranju.

Literatura

[1] Android Operating System, Wikipedia, The Free Encyclopedia (2012). Dostopno na: http://en.wikipedia.org/wiki/Android_%28operating_system%29.

[2] M. Gargenta, Learning Android, Sebastopol: O'Reilly Media, 2011, poglavja 1-3.

[3] Android - Mobilni Operacijski Sistem, Squidoo (2012). Dostopno na: http://www.squidoo.com/android-operacijski-sistem.

[4] Dashboards, Android developers (2012). Dostopno na: http://developer.android.com/about/dashboards/index.html.

[5] API Levels, Android developers (2012). Dostopno na: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels

[6] R. Meier, Professional Android 4 Application Development, Indianapolis: Wiley Publishing, 2012, poglavja 1-3.

[7] Fundamentals, Android developers (2012). Dostopno na: http://developer.android.com/guide/components/fundamentals.html.

[8] Eclipse software, Wikipedia, The Free Encyclopedia (2012). Dostopno na: http://en.wikipedia.org/wiki/Eclipse_%28software%29.

[9] N. Nedjah, L. Macedo Mourelle, System Enhineering using Particle Swarm Optimisation, New York: Nova Science Publishers, Inc., 2007, poglavje 4.

[10] J. Kennedy, R. Eberhart, Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks, 1995, poglavje 4, str. 1942-1948.

[11] I. Kononenko, M. Robnik-Šikonja, Inteligentni sistemi, Ljubljana: Založba FE in FRI, 2010, poglavje 12.3, str. 258-261.

[12] A.P. Engelbrecht, Computational Intelligence: An Introduction, England: Wiley Publishing, 2007, poglavje 4.

[13] XML, Wikipedia, The Free Encyclopedia (2012). Dostopno na: http://sl.wikipedia.org/wiki/XML

[14] Declaring Layout, Android developers (2012). Dostopno na: http://developer.android.com/guide/topics/ui/declaring-layout.html