SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3307

Otkrivanje statusa korisnika mobilnih uređaja

Nikola Goluban

Mentor: *Domagoj Jakobović*

Zagreb, lipanj, 2013.

Sadržaj

1. Uvod

U današnje vrijeme ljudi sve više koriste računala za komunikaciju, te sve većom rasprostranjenosti bežične tehnologije korisnici imaju mogućnost komunicirati sa svojim kontaktima bez obzira na mjesto i vrijeme gdje se oni ili njihovi kontakti nalaze. Naravno tu komunikaciju nije uvijek moguće ostvariti jer njihovi kontakti mogu biti zauzeti nekom drugom zadaćom. Sve većom uporabom pametnih telefona pojavljuje se i veća potražnja za informacijom je li određeni korisnik u mogućnosti se odazvati na neki zahtjev za komunikacijom.

Do sada uglavnom ne postoje metode kojima se automatski može odrediti takav status korisnika, ponajviše jer smo ograničeni kapacitetom baterije pametnog telefona. Ovaj rad će istražiti jedan od načina otkrivanja statusa korisnika mobilnog uređaja, uz dostupne podatke koje pametni telefoni danas pohranjuju, metodom strojnog učenja, točnije genetskim programiranjem.

2. Otkrivanje statusa korisnika mobilnih uređaja

2.1 Status korisnika i potreba za otkrivanjem statusa korisnika

U računalnim i telekomunikacijskim mrežama, prisutnost korisnika definiramo kao sposobnost i spremnost korisnika da komunicira. Danas korisnik može komunicirati preko niza uređaja s drugim korisnicima pa je zbog toga podatak o prisutnosti korisnika u danom trenutku bitan podatak za komunikaciju u stvarnom vremenu. Podatak o prisutnosti korisnika ima široku primjenu u mnogim komunikacijskim uslugama, kao što je *instant messaging* ili VoIP. Taj podatak informira ostale korisnike koji žele stupiti u kontakt s korisnikom o njegovoj prisutnosti. Uobičajena stanja prisutnosti su "slobodan za razgovor", "zauzet", "odsutan", "ne ometaj", "nedostupan".

U današnje vrijeme jedni od najčešćih korištenih uređaja za komunikaciju su pametni telefoni i oni sadrže mnoge usluge koje koriste podatke o prisutnosti korisnika. Iako često koriste podatak o prisutnosti, te imaju veliki izvor podataka vezanih za prisutnost korisnika, pametni telefoni ne koriste te podatke kako bi implicitno mijenjali prisutnost korisnika u skladu s njegovim kontekstom ili svakodnevnim ponašanjem. Razlog tomu je ograničenost baterije, kao i obilje podataka koje generiraju razni ugrađeni senzori i aplikacije, pa se informacija o prisutnosti uobičajeno mijena izravno intervencijom od strane korisnika ili implicitno, npr. kada se uređaj miruje duži vremenski period.

Komunikacijskim aplikacijama poput Skype-a ili GTalk-a, koji se danas sve više koriste na pamentim telefonima, informacija o prisutnosti korisnika vrši jednu od osnovnih uloga. Postojeće implementacije na pametnim telefonima podržavaju samo eksplicitnu promjenu statusa korisnika.

2.2 Automatsko ažuriranje statusa korisnika

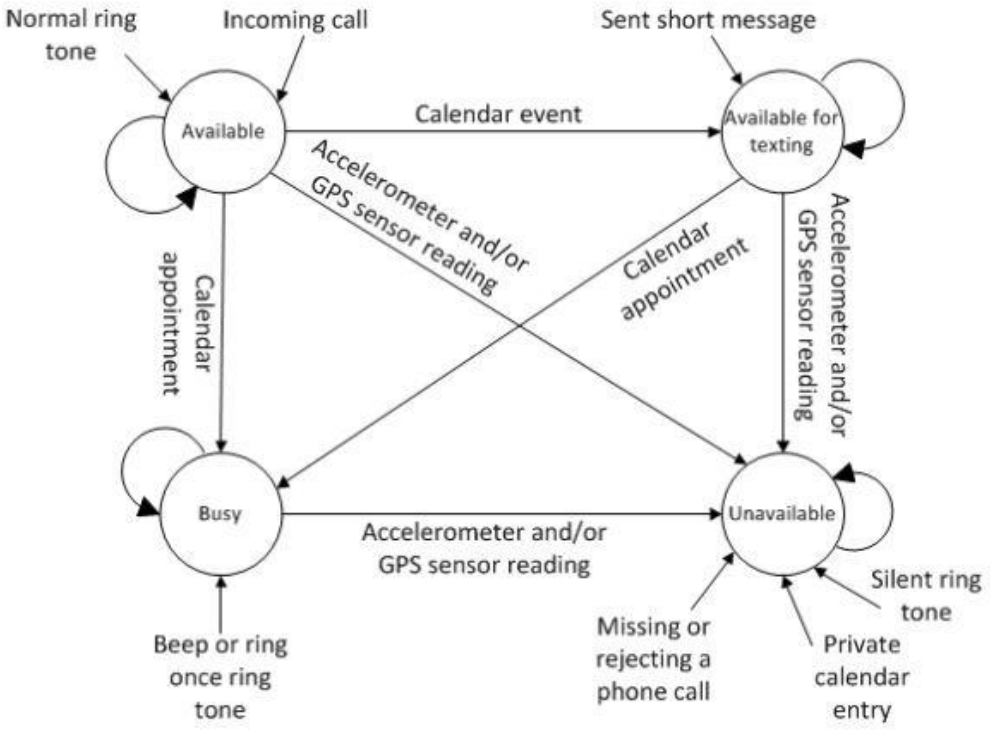
Zadaća automatskog ažuriranja statusa korisnika jest oslobađanje korisnika mobilnog uređaja od česte promjene statusa prisutnosti. Kao što je već navedeno pametni telefoni sadrže bogate informacije o prisutnosti korisnika, poput popisa poziva, kalendara događaja i sastanaka, zvuka zvonjave i sl. Na primjer, ukoliko je poziv odbijen u vrijeme zakazanog sastanka možemo zaključiti da je korisnik bio zauzet ili nedostupan. No ne vrijede ista pravila ponašanja za svakog korisnika. Korisnici imaju drugačije navike te prema tome za njih vrijede i drugačija pravila. Recimo neki korisnici ignoriraju pozive dok su na privatnom sastanku, dok će drugi možda biti slobodni za razgovor. Zbog toga automatsko ažuriranje korisnika treba odrediti prisutnost korisnika ovisno o osobnim obrascima ponašanja tog korisnika.

Kako bi ostvarili automatsko ažuriranje statusa korisnika prvo je potrebno odrediti stanja prisutnosti te akcije koje utječu na prisutnost korisnika.

Stanja prisutnosti:

1. Dostupan – stanje kada je korisnik sposoban i spreman komunicirati na bilo koji način
2. Nedostupan – stanje kada korisnik nije u stanju komunicirati
3. Dostupan za tekstualne poruke – stanje kada korisnik može komunicirati samo putem tekstualnih poruka
4. Zauzet – stanje kada korisnik trenutno ne može komunicirati

Akcije koje utječu na stanja prisutnosti možemo definirati sljedećim automatom:



Slika 1. Automat za dodjeljivanje statusa prisutnosti korisniku

Na primjer, ukoliko je korisnikov status bio „dostupan“ u trenutku kada je započeo neki događaj zabilježen u kalendaru tada je novi status korisnika „dostupan za tekstualne poruke“, ili ukoliko korisnik promjeni zvuk zvonjave u tiho bez obzira na prethodni status novi status korisnika je „nedostupan“.

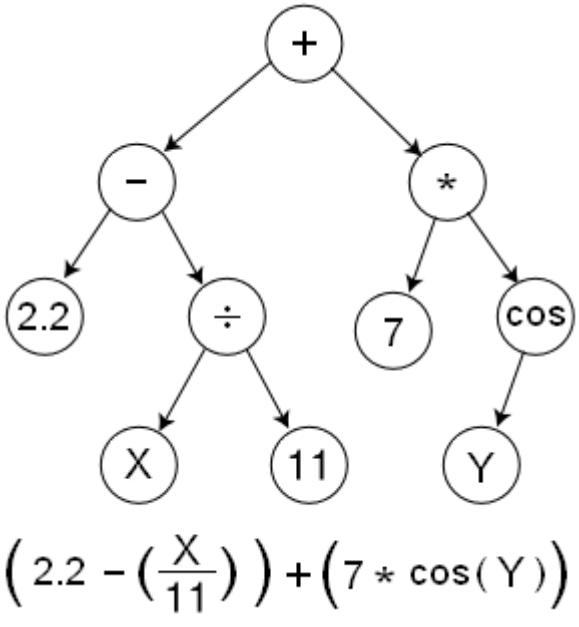
No promjena statusa korisnika mobilnog uređaja ne može se vršiti uvijek kada je nastupila neka od navedenih akcija. Ukoliko korisnik (ili neki od kontakata čiji status naša aplikacija prati) često mijenja status može isprazniti bateriju svog, odnosno tuđeg mobilnog uređaja. Zbog toga je potrebno smanjiti učestalost ažuriranja statusa korisnika ali pri tome zadržati prikladnu razinu točnosti.

3. Evolucijsko računanje

Evolucijsko računanje je naziv za tehnike rješavanja problema koje su inspirirane biološkom evolucijom. Ideja evolucijskog računanja je da nad populacijom rješenja primjenjuje evolucijske operatore (kao što su selekcija, križanje, mutacija itd.) koji populaciju postepeno vode k boljem rješenju. Evolucijsko računanje dijelimo na: genetske algoritme, genetsko programiranje, evolucijske strategije i evolucijsko programiranje.

3.1 Genetsko programiranje

Genetsko programiranje je tehnika evolucijskog računanja kod kojeg jedinka (jedno rješenje) predstavlja računalni program. Te programe najčešće prikazujemo kao stabla, jer se njima jednostavno prikazuju matematički i logički izrazi. Čvorovi stabala sadrže funkcijske znakove dok listovi sadrže varijable, konstante ili neke akcije. Genetski operatori koje genetsko programiranje koristi su: operator križanja i operator mutacije. Operator križanja uzima podstablo jednog roditelja i mijenja ga podstablom drugog roditelja, dok operator križanja može biti izveden na više načina, recimo može zamjeniti neki čvor stabla novim čvorom, može obrisati neki čvor, ili obrisati podstablo i zamijeniti ga novim podstablom i sl. Svaku jedinku evaluiramo pomoću funkcije dobrote, tj. funkcije koja nam govori koliko je dobro rješenje te jedinke.



Slika 2. Genetsko programiranje – program prikazan u obliku stabla

4. Programsko rješenje

Cilj ovog rada je istražiti mogućnost uporabe genetskog programiranja za otkrivanje statusa korisnika. Programsko rješenje je izvedeno uz pomoć Evolutionary Computation Framework (ECF) okvira za programski jezik C++ (neke funkcije korištene u programu su iz C++11 standarda), te programskog jezika python (v3.2). Podatci korišteni za učenje i testiranje su iz Mobile Data Challenge (MDC) skupa podataka prikupljenih tijekom kampanje prikupljanja podataka NRC – Lausanne (Lausanne Data Collection Campaign - LDCC), koja je dio Nokijine globalne istraživačke mreže, između 2009. i 2011.

4.1 Izvedba

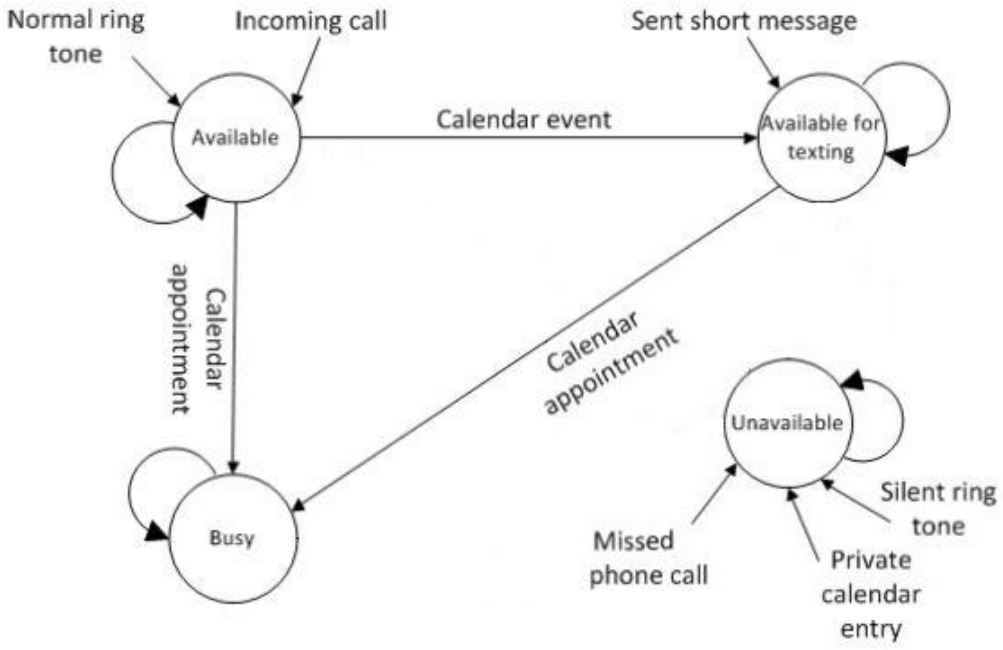
Rad se sastoji od 3 programa, od koja 2 pripremaju podatke za učenje i ispitivanje, dok treći uči genetski program (GP) te testira konačno dobiveno najbolje rješenje. Podaci uzeti za učenje i testiranje su od samo jednog korisnika, čime učimo GP navikama tog korisnika. Podaci koji su uzeti kao bitni za određivanje stanja korisnika su:

- Podaci o pozivu – smjer poziva (dolazni, odlazni ili propušteni), te vrsta poziva (glasovni poziv ili tekstualna poruka)
- Podaci kalendara – potvrđeni događaji i sastanci, te da li se radi o privatnom ili javnom događaju odnosno sastanku
- Podaci o zvuku zvona – da li je zvono postavljeno na normalni ili tihi rad
- Podaci o GSM mreži – identifikator mobilne mreže

Stanja korisnika su:

1. Dostupan
2. Dostupan za tekstualne poruke
3. Zauzet
4. Nedostupan

Iz tih podataka slijedi da se stanje korisnika određuje sljedećim automatom:



Slika 3. Automat za dodjeljivanje statusa korisnika

Treba primijetiti da neke akcije poput odlaznog poziva, dolazne poruke ili podataka o kodu mobilne mreže ne utječu na stanje u kojem se nalazi korisnik, no to ne znači da tim podacima ne možemo odrediti u kojem se stanju korisnik nalazi.

Izvedba ovakvog automata nije problem, no zbog limitiranosti baterije mobilnih uređaja ovakva izvedba bi trošila previše energije. Zbog toga ažuriranje stanja će se vršiti nakon određenog broja akcija koje pratimo. Problem koji zbog toga nastaje jest kako odrediti u kojem se stanju korisnik nalazio tijekom tih akcija, te kako procijeniti koja od tih akcija jest najviše utjecala na stanje kroz to vrijeme. Drugim riječima, želimo izbjeći provjeru svake akcije koja se dogodila u tom intervalu, kako bi uštedjeli na energiji i brzini izvođenja. U ovoj izvedbi status korisnika za određeni period jest status koji se najviše puta pojavio u tom periodu, npr. unutar 10 uzastopnih akcija za 4 akcije status korisnika bio je dostupan, za 3 akcije nedostupan, te za preostale 3 akcije zauzet, pa određujemo da za tih 10 akcija korisnih je bio dostupan. Kako ne bi provjeravali svaku akciju isto učinimo i za akcije, tako za primjer na 10 akcija u kojima su se pojavila dva odlazna poziva, dvije promjene zvuka zvona u normalni način zvona, po jedan dolazni poziv i jedna dolazna poruka, te četiri javna dogovora, za akcije uzimamo samo onu koja se najviše puta dogodila za određeni događaj, tj. za pozive uzimamo odlazni poziv, za zvuk zvona normalni zvuk, za unos u kalendaru uzimamo javni događaj.

4.2 Genetski program

Ostvareni program prikazan je kao stablo, pri čemu su mu svi čvorovi binarni, te provjeravaju logičke akcije, dok se u listovima nalaze konstante, odnosno predvidivi status korisnika. Svakim prolaskom kroz takvo stablo dobivamo stanje u kojem bi se korisnik trebao nalaziti za određene akcije. Funkcija dobrote se izračunava kao broj točno pronađenih statusa korisnika za zadane ulazne podatke.

Završni čvorovi:

- Dostupan
- Dostupan za tekstualne poruke
- Zauzet
- Nedostupan

Nezavršni čvorovi:

- Zvuk zvona – provjerava o kojoj vrsti zvonjave se radi. Može provjeravati da li se radi i tihom ili normalnom načinu rada.
- Vrsta poziva – provjerava o kojoj vrsti poziva se radi. Može provjeravati da li se radi o dolaznim, odlaznom, propuštenom pozivu ili o dolaznoj ili odlaznoj poruci.
- Zapis kalendara – može provjeravati da li se radi o javnom ili privatnom događaju ili o javnom ili privatnom sastanku
- Prethodni status – provjerava koji je status korisnika bio prije provjere novih akcija.
- Promjena mobilne mreže – provjerava da li je bilo promjena mreže prije prethodnih akcija

Nezavršni znakovi mogu vršiti provjeru samo za jednu od mogućnosti, tj. određeni čvor će vršiti provjeru da li je vrsta poziva bila odlazni poziv ali neće provjeravati da li se radi o dolaznom pozivu, za provjeru da li se radi o dolaznim pozivu brinut će se neki drugi čvor koji ce vršiti samo provjeru da li se radi o dolaznom pozivu. Za skup akcija uzimamo da je došlo do promjene mreže ako se mobilna mreža promijenila u barem 50% od ukupnih akcija.

4.3 Ulazni podaci

Ulazni podaci se koriste za učenje genetskog programa i za testiranje nakon što ga naučimo. Podaci koje koristimo nalaze se unutar četiri datoteke ("sys.csv", "calendar.csv", "calllog.csv", "gsm.csv"). Podatke prije učenja genetskog programa obradimo u programu napisanom u pythonu. Taj program uzima podatke iz navedene četiri datoteke te ih sortira ovisno u vremenu kada su se dogodili, također

treba napomenuti da ukoliko se unutar iste datoteke nalaze dva ili više zapisa koja su se dogodila u isto vrijeme samo onaj zadnji pročitani će se zabilježiti i obraditi. Tako sortirane podatke zapisuje u datoteku „zavrsniList.txt“, pri čemu kod mobilne mreže zapisuje u obliku „#kod“. Takvu datoteku dalje koristimo kako bi za sortirane podatke odredili status korisnika nakon svakog podatka. To činimo pomoću programa pisanog u c++-u (ime programa je „zavrsniInput“). Taj program učitava redom podatke iz datoteke i određuje stanje korisnika na temelju učitanog podatka, pri čemu se za početni status uzima status „dostupan“, te takve podatke sprema u strukturu koju potom zapisuje u formatiranu datoteku „inputData.bin“. Takvi podaci se učitavaju u program „zavrsni“ koji pomoću njih dalje uči i testira genetski program.

5. Rezultati

Genetski program učen je na skupu od 16000 akcija, pri čemu se evaluiralo po 80 uzastopnih akcija (što odgovara vremenskom razdoblju od 5 minuta za zadanog korisnika). Drugim riječima nakon 80 uzastopnih akcija određuje se status korisnika (onaj status koji se najviše puta pojavio kad bi smo određivali status za svaku akciju), takvih intervala od 80 akcija u skupu od 16000 akcija ima ukupno 200, pa je i maksimalna dobrota koju GP može ostvariti jednaka 200. Potom je GP testiran na skupu od 2000 akcija, koje su također grupirane po 80 akcija (maksimalna dobrota jest 25), pet puta.

5.1 Određivanje populacije i maksimalni broj generacija

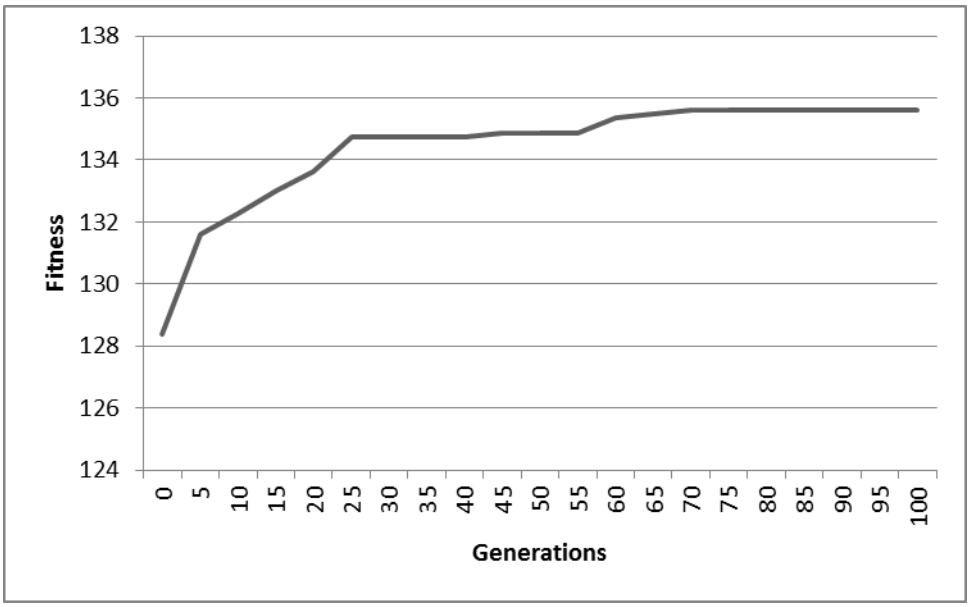
Veličinu populacije određena je tako da se je za 2000 evaluacija više puta učio i testirao GP za 20, 50, 100, 200, 400 jedinki pri maksimalnoj dubini stabla 3.

Tablica 1. Funkcija dobrote ovisno o veličini populacije

Populacija	*Dobrota – učenje*	*Dobrota – testiranje*
20	131.4	15.32
50	133.6	15.48
100	134	15.56
200	134	15.52
400	133.4	15.48

Kao što se iz tablice 1. može vidjeti za populaciju od 100 jedinki postigli smo najbolji uspjeh u učenju i testiranju, te za daljnje rezultate koristimo populaciju od 100 jedinki.

Maksimalni broj generacija određen je tako da je GP pušten više puta da uči stotinjak generacija. Iz grafa sa slike 4. može se vidjeti da nakon 25. generacije funkcija dobrote ne raste značajno (svega 0.875 na 50 idućih generacija, odnosno pronalazi jedno bolje rješenje). Stoga za maksimalni broj generacija na kojima učimo GP uzimamo 30 generacija.



Slika 4. Graf funkcije dobrote ovisno o broju generacija

5.2 Konačni rezultat

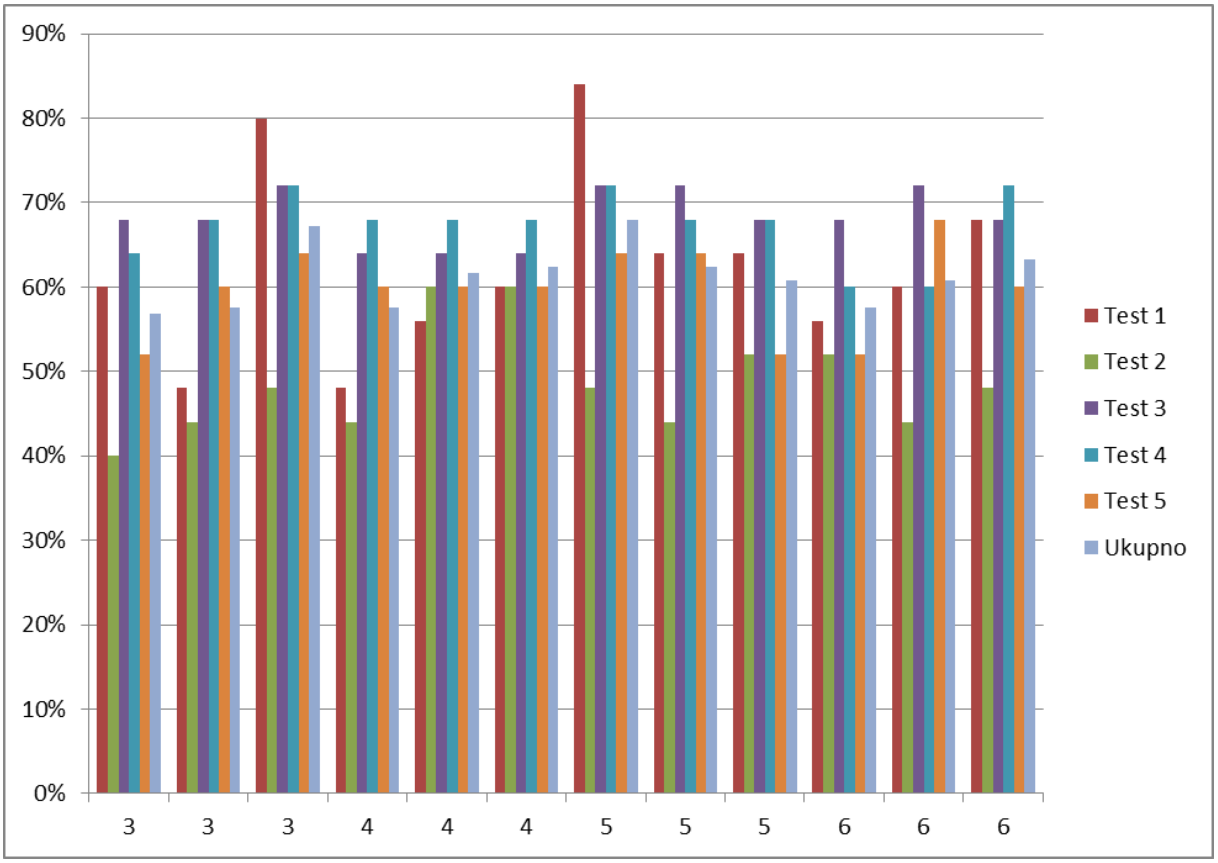
Jedini parametar koji nam ostaje za odrediti jest dubina stabla, no dubina stabla direktno utječe na brzinu izvođenja takvog programa čime se i utječe na potrošnju baterije. Zbog toga provjeravat ćemo dobivene programe za stabla dubine 3, 4, 5 i 6 te usporediti takve rezultate. Za svaku dubinu generirat ćemo 3 različita rješenja.

Tablica 2. Uspješnost GP-a na testnim primjerima

DUBINA	TEST 1	TEST 2	TEST 3	TEST 4	TEST 5	UKUPNO
3	60%	40%	68%	64%	52%	56.8%
3	48%	44%	68%	68%	60%	57.6%
3	80%	48%	72%	72%	64%	67.2%
4	48%	44%	64%	68%	60%	57.6%
4	56%	60%	64%	68%	60%	61.6%
4	60%	60%	64%	68%	60%	62.4%
5	84%	48%	72%	72%	64%	68%
5	64%	44%	72%	68%	64%	62.4%

5	64%	52%	68%	68%	52%	60.8%
6	56%	52%	68%	60%	52%	57.6%
6	60%	44%	72%	60%	68%	60.8%
6	68%	48%	68%	72%	60%	63.2%

Iz rezultata slijedi da GP pronalazi najbolja rješenja za dubinu stabla 5, no unatoč tome sa dubinom 3 pronašli smo jako dobro rješenje koje određuje točan status korisnika u 67.2% testnih primjera. Zbog toga što se radi o svega 3 dubine stabla takav program će se brzi izvoditi jer se mora obići manji broj čvorova da bi se došlo do rješenja.



Slika 5. Grafički prikaz uspješnosti pojedinih rješenja na testnim primjerima

6. Zaključak

Genetskim programiranjem možemo naučiti program da određuje prisutnost korisnika mobilnog uređaja, ali uz određeno odstupanje jer je nemoguće stvarno odrediti kada je korisnik voljan ili u mogućnosti da komunicira. Najveći problem pri rješavanju jest bio kako akcije koje utječu na status korisnika ili daju informaciju o statusu korisnika sažeti u nešto što će zahtijevati što manju procesnu snagu kako bi se u što manjoj mjeri trošila baterija.

U dobivenom programskom rješenju gdje smo koristili jednostavne funkcije koje samo provjeravaju da li se određena akcija dogodila, dobivamo rješenja koja se kreću oko 60% točnosti za testne podatke o korisniku. To daje naslutiti da ukoliko bi se koristile malo složenije funkcije, te točnijim određivanjem bitnih akcija u zadanom periodu, moglo bi se dobiti puno bolje rješenje za određivanje statusa korisnika koje bi bilo energetski učinkovito i dovoljno precizno za uporabu na mobilnim uređajima.

7. Literatura

[1] Aleksandar Antonić, Domagoj Jakobović, Ivana Podnar Žarko, „Inferring Presence Status on Smartphones: The Big Data Perspective“, http://www.fer.unizg.hr/_download/repository/iscc_2013_camera_ready.pdf, lipanj 2013.

[2] Wikipedia, „Presence information“, https://en.wikipedia.org/wiki/Presence_information, lipanj 2013.

[3] Wikipedia, „Evolutionary computation“, http://en.wikipedia.org/wiki/Evolutionary_computation, lipanj 2013.

8. Sažetak

U komunikacijskim uslugama, prisutnost korisnika je bitan podatak, kako je korisnik danas gotovo cijelo vrijeme povezan putem bežičnih mreža, taj podatak se često mora ažurirati. Danas ne postoje metode za primjenu na pametnim telefonima koje automatski ažuriraju status korisnika na temelju podataka koje mogu prikupiti, npr. popisu poziva, zapisa u kalendaru, podataka prikupljenih pomoću ugrađenih senzora itd. Zadaća automatskog ažuriranja statusa jest da oslobodi korisnika od eksplicitne promjene statusa korisnika svaki puta kad je on voljan ili u mogućnosti da komunicira.

Ovim radom se je istražila mogućnost pronalaska programa koji bi određivao status korisnika na temelju dostupnih podataka pomoću genetskog programiranja. Status korisnika određuje automatom sa slike 3. Zbog ograničenosti baterije i velike količine podataka koja se često generira ne možemo za svaku akciju provjeravati i mijenjati status korisnika već to činimo tako da za niz akcija odaberemo one koje su se najčešće pojavljivale, a pri tome će se status odrediti kao onaj u kojem se je korisnik najčešće nalazio.

Rezultati rada su davali točnost od oko 60% na 5 testnih primjera, pri čemu nije bilo potrebno raditi velika stabla programa kako bi se dobili dobri rezultati.

Ključne riječi: strojno učenje, genetsko programiranje, otkrivanje statusa korisnika mobilnih uređaja

9. Summary

In the communication services, user presence status is essential information. Nowadays users are connected almost all the time to other users through wireless networks, therefore presence information often needs to be updated. Today there are no methods for use on smartphones that can automatically update user status based on the information which they collect, e.g. call log, calendar entries, data collected by using built-in sensors etc. Task of automatic status update is to free users from explicit user status changes whenever he is willing and able to communicate.

This paper has examined the possibility of finding program that would determine user status based on the available data using genetic programming. User status is determined by a slot machine from the Figure 3. Due to limited battery life and a large amount of data that is often generated we cannot check and change user status for each action, instead for a number of actions we choose those that are most frequently occurred, and at the same time status is defined as one in which the user is most often found.

Results of this research gave accuracy of about 60% to 5 test cases, in which there was no need to do a large tree structure of the program in order to get good results.

Keywords: machine learning, genetic programming, discovering the status of mobile users