ZBATIMI I ALGORITMIT GJENETIK NË ZGJIDHJEN E PROBLEMIT "BIZNESMENI UDHËTAR" DHE PROGRAMIMI I PAJISJEVE ME SISTEM OPERATIV ANDROID

PhD(c) Laurik Helshani

Dorëzuar
Universitetit Europian të Tiranës
Shkollës Doktorale

Në përmbushje të detyrimeve të programit të Doktoratës në
Ekonomi & IT me profil Menaxhim i Sistemeve të Informacionit, për marrjen e gradës shkencore "Doktor"

Udhëheqës shkencor: PhD Ervin Ramollari

Numri i fjalëve: 50 412

Tiranë, Janar 2018

DEKLARATA E AUTORËSISË

Unë, Laurik Helshani, deklaroj që disertacioni im është punë origjinale e imja. Pra, deklaroj që nuk kam përdorur ndonjë ide, të dhënë apo gjuhë të një subjekti tjetër pa ia atribuuar autorësinë dhe pa vendosur referencën specifike.

Po ashtu, deklaroj se pjesë të veçanta të programimit dhe të të dhënave nga ky punim, janë përdorur për konferenca dhe botime në revista shkencore, në përmbushje të obligimeve të studimit doktoral.

ABSTRAKTI

Te problemi biznesmeni udhëtar, biznesmeni planifikon një tur nëpërmjet të cilit dëshiron t'i vizitojë gjithë klientët e tij. Gjatë udhëtimit asnjë nga klientët nuk duhet të anashkalohet si dhe asnjëri prej tyre nuk duhet të vizitohet më shumë se një herë. Pas mbarimit të turit biznesmeni udhëtar duhet të kthehet përsëri në vendin e nisjes dhe gjatësia e rrugës së turit duhet të jetë sa më e shkurtër dhe si rrjedhojë më pak e kushtueshme.

Modelimi matematik i këtij problemi bëhet përmes grafeve, ku nyjet e grafit simbolizojnë vendet për vizitë, ndërsa brinjët e grafit që lidhin nyjet e tij simbolizojnë rrugët që çojnë tek ato vende. Kompleksiteti i algoritmeve ekzakte për zgjidhjen e këtij problemi rritet me rritjen e numrit të vendeve për vizitë. Për këtë arsye, në këtë studim përzgjidhet algoritmi gjenetik për optimizimin e rrugës. Ky algoritëm nuk garanton zgjidhje ekzakte, mirëpo ofron zgjidhje të përafërt dhe të kënaqshme.

Në këtë studim, përmes simulatorëve të programuar në Java, prezantohet grafikisht puna e algoritmit gjenetik si dhe atij simulated annealing, gjatë zgjidhjes së problemit biznesmeni udhëtar. Po ashtu është bërë krahasimi i rezultateve të përftuara nga simulatorët për ta testuar performancën e tyre: optmizimin e rrugës dhe kohëzgjatjen e ekzekutimit të tyre. Rezultatet tregojnë se algoritmi simulated annealing ka performancë më të mirë se algoritmi gjenetik, meqë ai është më i shpejtë dhe optmimizon rrugën më tepër se algorimi gjenetik.

Një qëllim tjetër i këtij studimi është aplikimi i zgjidhjes së këtij problemi në planifikimin e udhëtimeve nëpër një numër vendesh gjeografike me koordinata të dhëna. Llogaritja e distancave (kostove) mes dy pikave është më komplekse se sa bashkimi i tyre në vijë të drejtë, për shkak të formës së tokës (sferë e shtypur në pole) dhe të rrugëve jo të drejta që mund të kalojnë edhe nëpër vende të tjera gjeografike. Për t'u marrë me këtë problematikë, autori është nxitur gjithashtu nga dëshira për të programuar një aplikacion që t'i kontribuojë turizmit në trevat shqiptare, duke u lehtësuar vizitorëve të huaj gjetjen më të lehtë të vendeve me vlerë historike, bukurive natyrore etj.

Sistemi softuerik që është product final i punës praktike së kësaj teze doktorale, është programuar bazuar në arkitekturën e orientuar nga shërbimet. Algoritmi gjenetik ofrohet si ueb shërbim RESTful nga ana e serverit, i cili konsumohet nga aplikacioni Android si klient, për optimizimin e rrugës. Përveç kësaj aplikacioni i Androidit integron ueb shërbimet e Google-it në formë hartash, përdor ato për vizatimin e rrugës mbi hartën e Google-it, për të caktuar distancat në mes të vendeve për vizitë si dhe të ofrojë një bashkëveprim më të madh me përdoruesin.

Ky studim kombinon në vete fusha të ndryshme si: teorinë e grafeve nga matematika diskrete, algoritmet që zbatohen mbi grafe nga inxhinieria e algoritmeve dhe programimi i pajisjeve mobile me sistem operativ Android.

Fjalët kyçe: *Travelling Salesman Problem (TSP), Algoritmi gjenetik (GA), Optimizimi, Android, Google Maps, ueb shërbimet RESTful, Java*

ABSTRACT

In the Travelling Salesman Problem, a salesman plans a tour through which he intends to visit all of his clients. During the trip, no client should be skipped and no client should be visited more than once. In the end of the trip, the travelling salesman should return to the starting point and the length of the tour should be as short and as less costly as possible.

The mathematical modeling of this problem is done by means of graphs, where the nodes of the graph represent the places to be visited, while the edges of the graph connecting the nodes represent the roads that lead to those places. The complexity of exact algorithms in solving this problem increases with the number of places to visit. For this reason, the genetic algorithm has been chosen by the author for the optimization of the trip. This algorithm does not guarantee an exact solution, but it offers an approximate and satisfactory solution.

This study, by means ofJava simulators visually presents the works of the genetic algorithm and the simulated annealing algorithm as they solve the travelling salesman problem. Also, a comparison of results obtained from the simulators has been performed, in order to test their performance: trip optimization and execution time. The results show that simulated annealing performs better than the genetic algorithm, since it is faster and optimizes the trip more than the genetic algorithm.

Another aim of this study has been to apply the solution to this problem to the planning of trips through a number of geographic places with given coordinates. The calculation of distances (costs) between any two points is more complex than their connection with a straight line, because of the Earth's shape (sphere squeezed in the poles) and the non-straight roads that may pass through other geographic places as well. In studying this problem, the author has been driven by the desire to program an application that will contribute tourism in the Albanian territories, thus making it easier for foreign visitors to find places of historic values, natural points of attraction, etc.

The software system, which is the final product of the practical work in this doctoral thesis, has been programmed based on the Service Oriented Architecture. The genetic algorithm has been offered as a RESTful web service from the server side, which is then consumed by the Android application as a client, for optimizing the trip. Besides, the Android application integrates Google web services for maps, uses them to draw the roads on Google Maps, for finding distances between visited places, as well as to offer a better user interaction.

The research presented in this thesis combines different areas, such as: graph theory from discrete maths, algorithms on graphs from algorithmics, and programming of mobile devices with Android OS.

Keywords: *Travelling Salesman Problem (TSP), Genetic Algorithm (GA), Optimization, Android, Google Maps, RESTful web services, Java*

FALËNDERIME

Falënderoj Zotin që më dha shëndetin, mendjen dhe aftësinë intelektuale.

Në radhë të dytë falënderoj familjen time të ngushtë për përkrahjen morale, si dhe udhëheqësin dhe mentorin tim doktoral Prof. PhD Ervin Ramollarin për gatishmërinë e tij, këshillat dhe udhëzimet që më dha gjatë shkrimit të këtij disertacioni.

PËRMBAJTJA E LËNDËS

LISTA E TABELAVE x
LISTA E FIGURAVE xi
LISTA E SHKURTIMEVE DHE E FJALORIT xii
KAPITULLI I: HYRJA 1
1.1. Algoritmi Gjenetik 6
1.2. Pyetjet kërkimore 6
1.3. Objektivat e studimit 7
1.4. Hipoteza 7
1.5. Motivimi për të studiuar problematikën e zgjedhur 8
1.6. Mundësitë dhe kufizimet 8
1.7. Rëndësia e studimit dhe kontributi i tij 10
1.8. Metodat dhe metodologjia e kërkimit 10
1.9. Struktura e punimit doktoral 11
KAPITULLI II: PROBLEMI BIZNESMENI UDHËTAR DHE HULUMTIME TË NGJASHME 15
2.1. Historiku 15
2.2. Origjina e problemit 16
2.3. Definimi i problemit 17
2.4. Klasifikimi i problemit 17
2.5. Konceptet themelore të teorisë së grafeve 17
2.6. Formulimi matematik i problemit 19
Formulimi përmes grafeve 19
Formulimi si programim linear 19
2.7. Aplikimet e TSP-së dhe problemet e ngjashme me të 20
Kabllimi në kompjuter (Computer wiring) 20
Rrugëtimi i automjeteve-kamionëve (Vehicle Routing) 21
Rrugëtimi i autobusëve shkollorë 21

Design of global navigation satellite system surveying networks (GNSS) 21
Procesi i shpimit te pllakat elektronike 22
2.8. Metodat numerike (algoritmet) për zgjidhjen e problemit TSP 23
Algoritmet ekzakte 23
Algoritmet joekzakte 30
2.9. Punime krahasuese bazuar në rezultate eksperimentale 48
Krahasim i precizitetit në mes OpenStreetMap, Bing Maps dhe Google Maps në lidhje me Irlandën 58
2.10. Përmbledhje 60
KAPITULLI III: ARKITEKTURA E ORIENTUAR DREJT SHËRBIMEVE (SOA) 62
3.1. Hyrje 62
3.2. Komponentët arkitekturore të SOA-së 63
3.3. Ueb shërbimet 64
Shtresat e teknologjisë së ueb shërbimeve 65
Arkitektura e ueb shërbimeve 67
SOAP 69
UDDI – Universal Discovery, Description Language 73
Web Service Description Language (WSDL) 74
Representational State Transfer REST 78
Hyper Text Transfer Protocol (HTTP) 81
3.4. Ueb shërbimet e Google-it 82
Pasqyrë e përgjithshme 82
Google Maps Distance Matrix API 83
Google Maps Distance Matrix – Përgjigjja 87
Google Maps Directions API 88
Google Maps API 93
The Google Maps Geocoding API 95
Google Places API 97
Krahasimi mes Google Maps dhe Bing Maps 99
Krahasimi mes Open Street Map dhe Google Maps 104

3.5. Kompjuterizimi në Re 106
Përparësitë e Kompjuterizimit në Re 108
Modelet e shërbimit të Kompjuterizimit në Re 111
Modelet e shpërndarjes së shërbimeve të Kompjuterizimit në Re 112
Lidhja midis SOA-s dhe Kompjuterizimit në Re 116
Kompjuterizimi në Re dhe virtualizimi 116
Teknikat që kontribuojnë në teknologjinë e reve kompjuterike 117
Virtualizimi 119
Barrierat në retë kompjuterike 129
Rreziqet në retë kompjuterike 130
Çështjet etike të Kompjuterizimit në Re 132
3.6. Kompjuterizimi në Re mobil (Mobile Cloud Computing MCC) 133
Përfitimet që sjellë MCC 133
Pikat e dobëta të MCC-së 133
Arkitektura e MCC-së 134
Arkitektura e aplikacioneve MCC 134
Shërbimet tipike që nevojiten nga një server MCC 135
Përparësitë e MCC-së 136
Aplikimet e Mobile Cloud Computing 139
3.7. Përparësitë e SOA-së 143
3.8. Sfidat e SOA-së 144
3.9. Përmbledhje 144
KAPITULLI IV: PROGRAMIMI I SISTEMIT TË PROPOZUAR 150
4.1. Programimi i algoritmit gjenetik si ueb shërbim 150
Simulimi i evolucionit në kompjuter 152
Zbatimi i algoritmit gjenetik në zgjidhjen e problemit TSP 154
Parametrat që kontrollojnë punën e algoritmit gjenetik 155
Implementimi i algoritmit gjenetik (përbërja dhe shpjegimi i klasave) 156
Turi 162
Kriteret e ndërprerjes së punës të algoritmit gjenetik 166

Ofrimi i algoritmit gjenetik si ueb shërbim RESTful (Web-Service)..... 167
4.2. Programimi i aplikacionit për pajisje mobile me sistem operativ Android..... 170
Hyrje në sistemin operativ Android..... 170
Programimi i Android-Manifestit..... 175
Komente në lidhje me skedarin konkret të manifestit të aplikacionit..... 177
Programimi i preferencave (parapëlqimeve)..... 179
4.3. Përmbledhje..... 189
KAPITULLI V: ANALIZË, TESTIM DHE INTERPRETIM REZULTATESH..... 195
5.1. Interpretim rezultatesh të algoritmit gjenetik..... 197
5.2. Interpretim rezultatesh të algoritmit Simulated Annealing..... 199
5.3. Testimi i të gjithë sistemit (punës praktike të doktoratës)..... 201
Karakteristikat e mjedisit për testim..... 201
Ecuria e testimit të të gjithë sistemit (cikli komplet)..... 202
Interpretimi i rezultateve..... 210
KAPITULLI VI. PËRFUNDIME/REKOMANDIME..... 214
6.1. Puna në të ardhmen..... 218
SHTOJCAT..... 220
Shtojca A: Klasa JsonReader..... 220
Shtojca B: Klasa Vendi..... 224
Shtojca C: GooglePlacesAutocompleteAdapter..... 229
Shtojca D: PlaceAdapter..... 232
Shtojca E: Directions dhe polylines..... 234
REFERENCAT..... 238

LISTA E TABELAVE

Tabela 1: Pseudocode i Simulated Annealing..... 31
Tabela 2: Forma e përgjithshme e algoritmit gjenetik 32
Tabela 3: Parametrat e përdorur nga të dy algoritmet (gjenetik dhe simulated annealing) 55
Tabela 4: Shembull nga kodi XML..... 68
Tabela 6: Skema e XML-it..... 69
Tabela 7: SOAP-Mesazhi..... 71
Tabela 8: Kërkesë-përgjigje e një funksioni në WSDL 78
Tabela 9: Përgjigjja e ueb shërbimit Distance Matrix si JSON 86
Tabela 10: Përgjigjja e ueb shërbimit Distance Matrix si XML 86
Tabela 11: Lista e gjuhëve në dispozicion të përgjigjes së shërbimeve..... 90
Tabela 12: Përgjigjja e ueb shërbimit geo-code në JSON..... 97
Tabela 13: Open Street Map vs Google Maps 106
Tabela 14: Ndërlidhje konceptesh mes TSP-së dhe teorisë së grafeve..... 152
Tabela 15: Domethënia e termave biologjikë tek algoritmi biologjik 154
Tabela 16:Programimi i formulës së Haversinës 158
Tabela 17: Programimi i funksionit për llogaritjen e gjatësisë së turit duke përdorur Google Distance Matrix..... 161
Tabela 18: Pjesë nga kodi i ueb shërbimit RESTful 169
Tabela 19: Versionet e sistemit operativ Android dhe karakteristikat e tij..... 174
Tabela 20: Programimi i Manifestit të Androidit përmes XML-it..... 177
Tabela 21: Programimi i ndërfaqes së preferencave përmes XML-it..... 180
Tabela 22: Programimi i funksionalitetit të preferencave me JAVA..... 183
Tabela 23: Programimi i event-it onMapLongClick..... 186
Tabela 24: Funksioni HTTP - POST (JAVA)..... 188
Tabela 25: Rastet për testim dhe rezultatet e algoritmit gjenetik..... 197
Tabela 26: Rastet për testim dhe rezultatet e algoritmit Simulated Annealing..... 200
Tabela 27: Testi I me 3 vende për t'u vizituar 205
Tabela 28: Testi II me 4 vende për t'u vizituar..... 206
Tabela 29: Testi III me 8 vende për t'u vizituar..... 207
Tabela 30: Testi IV me 12 vende për t'u vizituar 208
Tabela 31: Testi V me 16 vende për t'u vizituar..... 209
Tabela 32: Parametrat kryesorë që ndikojnë në kohëzgjatjen e sistemit (kohëzgjatja në sekonda) 210

LISTA E FIGURAVE

Figura 1: Analogjia e algoritmit të kolonisë së milingonave me atë si funksionojnë në natyrë 43
Figura 2: GUI i algoritmit gjenetik dhe kolonisë së milingonave 53
Figura 3: Shtresat e teknologjisë së ueb-shërbimeve (Tidwell, Snell, & Kulchenko, 2001) 65
Figura 4: Struktura e SOAP-mesazhit 71
Figura 5: Struktura e WSDL-dokumentit 77
Figura 6: Kompjuterizimi në Re (Musafi) 107
Figura 7: Arkitektura e Virtualizimit (Kalavace, 2013) 121
Figura 8: Ruajtja e të dhënave ne Cloud 138
Figura 9: Shembuj nga Cloud 138
Figura 10: Tregtia mobile 140
Figura 11: Të mësuarit mobil 141
Figura 12: Kujdesi shëndetësor mobil 142
Figura 13: Lojëra mobile 143
Figura 14: Principi i punës së algoritmit gjenetik 153
Figura 15: Diagrami i klasave të algoritmit gjenetik 167
Figura 16: Komunikimi mes pajisjes inteligjente dhe serverit TOMCAT 169
Figura 17: GUI i preferencave 184
Figura 18: Pamje nga aplikacioni i androidit gjatë përzgjedhjes së endeve për vizitë përmes kërkimit 187
Figura 19: Rruga e optimizuar në aplikacionin klient të Androidit 189
Figura 20: Simulatori për testimin e algoritmit gjenetik 196
Figura 21: Simulatori për testimin e algoritmit Simulated Annealing 200
Figura 22: Parametrat kryesorë që ndikojnë në kohëzgjatjen e sistemit (grafikisht) 211

LISTA E SHKURTIMEVE DHE E FJALORIT

HTML- HyperText Markup Language

JSON - JavaScript Object Notation

XML- EXtensible Markup Language

WSDL - Web Services Description Language

SOAP - Simple Object Access Protocol

HTTP - Hypertext Transfer Protocol

UDDI - Universal Description, Discovery and Integration

DTD - Document Type Definition

REST - Representational State Transfer

TCP/IP –Transmission Control Protocol/Internet Protocol

GUI - Graphical User Interface

TSP – Travelling Salesman Problem

GNSS - Design of global navigation satellite system surveying networks

SOA - Service Oriented Architecture

CC – Cloud Computing

MCC - Mobile Cloud Computing

RGESA - Randomized Gravitational Emulation Search Algorithm

OSM – Open Street Map

KAPITULLI I: HYRJA

Udhëtimi në ditët e sotme nuk është më sfidë, mirëpo sfidë paraqet optimizimi (shkurtimi) i rrugës së udhëtimit, dhe si rrjedhojë ka kursimin e kohës dhe të parave (ulja e shpenzimeve të karburantit). Edhe pse ekzistojnë metoda numerike ekzakte për zgjidhjen e këtij problemi, kohëzgjatja e ekzekutimit të tyre rritet në faktorial (n!) me rritjen e numrit të vendeve për vizitë. Ky hulumtim doktoral do të përqendrohet të metodat numerike që ofrojnë zgjidhje të përafërta për optimizim.

Me kombinimin e metodave të ndryshme numerike, teknologjive të ndryshme që zbatohen në Internet, si dhe pajisjeve të mençura mobile kjo sfidë do të tejkalohet.

Trendet e sotme teknologjike, bazuar në standarde dhe protokolle të ndryshme, që zbatohen në Internet, janë duke shkuar drejt komunikimit në distancë. Për këtë arsye ueb shërbimet shihen si teknologjia e duhur që mundëson ekzekutimin e funksioneve në largësi dhe në njëfarë mënyre e prezantojnë softuerin si shërbim, ku theks i veçantë i vihet sigurisë gjatë shkëmbimit të informacioneve në Internet.

Google është gjiganti që ofron ueb shërbime interaktive në formë hartash për përdorues të ndryshëm. Këto ueb shërbime do të përdoren si mjet për të arritur te qëllimi, në mënyrë që aplikacioni përfundimtar të jetë sa më ndërveprues dhe më intuitiv (i vetëkuptueshëm) për përdoruesin.

Qëllimi i këtij punimi doktoral nuk është vetëm vërtetimi i hipotezës duke bërë që doktoratura të mbetet vetëm kontribut intelektual, siç kanë bërë shumë doktorantë të

mëparshëm, por edhe të aplikohet duke zhvilluar një produkt (softuer) i cili mund të ndihmojë turizmin në vendet tona. Këtë e kam bërë duke futur në lojë pajisjet e mençura (smart) me theks të veçantë smart-phone-ët me sistem operativ Android, meqë vitet e fundit ato janë bërë pjesë e pandashme e jetës sonë për të gjitha grup-moshat. Vizitori i huaj që ka për synim të vizitojë vendet tona, qofshin ato bukuri natyrore ose vende me vlerë historike, mjafton ta shkarkojë dhe ta instalojë aplikacionin dhe në hartat e Google-it mund të zgjedhë vendet për vizitë. Vizitori pret që aplikacioni ta kryejë punën e tij dhe pasi merr si rezultat rrugën e vizatuar në hartë, pra rrugën më të shkurtër dhe më pak të kushtueshme, që të çon tek të gjitha vendet e dëshiruara të kthen prapë në pikën e nisjes.

I gjithë sistemi, i cili do ta jetësojë punimin tim doktoral, do të bazohet në arkitekturën tashmë të njohur klient-server, ku algoritmi gjenetik ofrohet si ueb shërbim RESTful, ndërsa ana e klientit do të jetë prapë një softuer i aplikueshëm në pajisjet me sistem operativ Android. Algoritmi gjenetik simulon idenë e evolucionit në kompjuter, ku më i forti mbijeton. Ky sistem është programuar në atë mënyrë që të integrojë shërbimet e Google-it, por edhe ta konsumojë ueb shërbimin RESTful përmes të cilit ofrohet i gjithë algoritmi gjenetik si i tillë.

Arsyet pse u përzgjodh që algoritmi gjenetik të ofrohet si ueb shërbim RESTful e jo si SOAP janë të shumta: stili arkitekturor, përdorimi i protokollit HTTP, kthimi i përgjigjes në formate të ndryshme, shpejtësia, bandwidth, pavarësia e transmetimit, identifikimi i burimeve, siguria, caching dhe qasja. Këto arsye si edhe krahasimet e mëposhtme janë të shpjeguara në (Difference between SOAP and RESTful Web Service in Java, 2015):

(i) REST është një stil arkitekturor, në të cilin janë të ndërtuara ueb shërbimet RESTful, ndërsa SOAP është një standard i hartuar për të përmirësuar komunikimin në mes të klientit dhe serverit për sa i përket formatit, struktura dhe metoda.

(ii) REST shfrytëzon protokollin HTTP, duke përfshirë metodat p. sh.: GET, POST, PUT dhe DELETE për të ndërmarr ndonjë veprim p. sh.: nga një aplikacion i cili ofron të dhënat në lidhje me librat, kërkesa GET mund të përdoret për të thirrur librat, POST mund të përdoret për të ngarkuar të dhënat e një libri të ri, dhe DELETE mund të përdoret për ta fshirë një libër nga biblioteka. Nga ana tjetër, SOAP përdor mesazhet XML për të komunikuar me server.

(iii) Ueb shërbimet RESTful mund të kthejnë përgjigje në formate të ndryshme p. sh.: JSON, XML dhe HTML, ndërsa duke përdorur ueb shërbimin SOAP që lidhin përgjigjen me XML për shkak se përgjigjja e vërtetë është e paketuar brenda një mesazh SOAP i cili është gjithmonë në formatin XML.

(iv) Përpunimin e një kërkesë të një ueb shërbimet RESTful, që është shumë më i shpejt sesa përpunimi i një mesazhi SOAP, meqë për këtë të fundit nevojitet parsim (përpunim). Për këtë arsye ueb shërbimet RESTful janë më të shpejta sesa shërbimi SOAP.

(v) Mesazhet SOAP konsumojnë më shumë bandwidth sesa mesazhet RESTful për të njëjtin lloj të operacionit, sepse XML është më fjalëshumë sesa JSON. Mënyra standarde për të dërguar mesazhe SOAP ka një kokë shtesë për çdo mesazh, ndërsa ueb shërbimet RESTful shfrytëzojnë HTTP - header.

(vi) Meqë mesazhet SOAP janë të mbështjella në një zarf (pliko) SOAP ato mund të dërgohen përmes çdo mekanizmi të transportit p. sh.: TCP, FTP, SMTP ose me ndonjë protokoll tjetër. Nga ana tjetër, ueb shërbimet RESTful janë shumë të varura nga protokolli HTTP. Ato përdorin urdhra HTTP, funksionimin e tyre dhe varen nga HTTP për transmetimin e përmbajtjes në server. Edhe pse në një botë reale, SOAP funksionon më së shumti përmes HTTP-s, mund të thuhet që ky avantazh i pavarësisë së transportit nuk është shfrytëzuar realisht.

(vii) Ueb shërbimet RESTful përdorin URL-n për të identifikuar burimet e dëshiruara që do të arrihen, ndërsa SOAP përdor mesazhet XML për identifikimin e ueb procedurave dhe burimeve, që duhen thirrur apo konsumuar.

(viii) Siguria në ueb shërbimet RESTful mund të zbatohet duke përdorur zgjidhje standarde dhe tradicionale për qasje të autorizuar në burime të caktuara të internetit. Ndërsa për zbatimin e sigurisë në ueb, shërbimet e internetit SOAP duhet të kenë infrastrukturë shtesë në ueb për të mundësuar mesazhet ose çështjet e sigurisë në nivel transporti.

(ix) Ueb shërbimi RESTful ka avantazh të plotë në përdorimin e mekanizmit web - cashing, për shkak se në thelb është i bazuar në URL. Nga ana tjetër ueb shërbimi SOAP e injoron plotësisht mekanizmin web – caching.

(x) Tek ueb shërbimet RESTful çdo entitet është i përqendruar rreth burimeve, ndërsa në rastin e ueb shërbimeve SOAP, çdo entitet është i përqendruar rreth ndërfaqeve dhe mesazheve.

Ajo çka e bën këtë punim doktorature më të veçantë krahasuar me punimet e tjera është se ky punim kombinon në vetvete disa fusha të ndryshme: teorinë e grafeve nga matematika diskrete, inxhinierinë e algoritmeve dhe programimin e pajisjeve me sistem operativ android me theks të veçantë telefonat e mençur. Ofrimi i algoritmit gjenetik si ueb shërbim RESTful, futja në lojë e Androidit, duke përfshirë këtu integrimin e ueb shërbimeve të Google-it dhe përdorimi i algoritmit gjenetik si ueb shërbim për optimizimin e rrugës janë disa nga pikat që e bëjnë punimin tim doktoral të dallojë nga të tjerët.

Në epokën tonë udhëtimi është i domosdoshëm. Nuk bëhet më fjalë të shkosh prej një vendi në tjetrin, meqë mundësitë dhe mjetet e udhëtimit janë të shumta, mirëpo sfida është shkurtimi (optimizimi) i kësaj rruge.

Një shitës (biznesmen) planifikon një udhëtim përmes të cilit dëshiron t'i vizitojë klientët e tij dhe të kthehet prapë në pikën e nisjes. Gjatë rrugës vajtje-ardhje klienti i njëjtë nuk duhet të vizitohet më tepër se një herë, por edhe asnjë klient nuk duhet lënë pa u vizituar: biznesmeni duhet të kthehet në pikën e nisjes, dhe rruga duhet të jetë sa më e shkurtër dhe sa më pak e kushtueshme. Modelimi matematik i këtij problemi ka të bëjë me teorinë e grafeve dhe kombinatorikën. Nyjet e grafit simbolizojnë vendet për vizitë, ndërsa brinjët që lidhin nyjet e tij janë rrugët (gjatësia e të cilave dihet nga fillimi) që të çojnë tek ato vende.

Kompleksiteti i algoritmit ekzakt për zgjidhjen e këtij problemi rritet me rritjen e numrit të vendeve për vizitë në faktorial (n!). Për këtë arsye ofrohet algoritmi gjenetik si zgjidhje optimale e kënaqshme e problemit TSP. Principi i punës së këtij algoritmi bazohet

në teorinë e evolucionit, respektivisht në përzgjedhjen natyrore (ku më i forti mbijeton). Pra, ka të bëjë me gjenetikën nga shkenca e biologjisë.

1.1. Algoritmi Gjenetik

Thelbi i principit të punës së algoritmit gjenetik është ndryshimi i bashkësisë së zgjidhjeve të mundshme hap pas hapi. Gjatë ndryshimeve zgjidhjet shumë të mira ruhen, ndërsa ato shumë të këqijat, me gjasë shumë të madhe, hidhen poshtë dhe kështu në bazë të variacioneve dhe kombinacioneve krijohen zgjidhje të reja. Tek algoritmi gjenetik zgjidhjet e vetme trajtohen si individ, ndërsa bashkësia e zgjidhjeve si popullsi. Gjatë ekzekutimit të algoritmit, nga epoka në epokë popullsia fillestare ndryshohet. Këto ndryshime pasojnë nga rikombinimi dhe mutacioni i materialit gjenetik. Thënë shkurt, përmes algoritmit gjenetik simulohet evolucioni në kompjuter, pra seleksionimi natyror ku më i forti mbijeton.

1.2. Pyetjet kërkimore

Pyetja I: Sa ndikon algoritmi gjenetik në optimizimin e rrugës te problemi biznesmeni udhëtar?

Pyetja II: Cili nga algoritmet optimizon rrugën më tepër dhe jep rezultate në kohë më të shkurtër: ai gjenetik apo ndonjë algoritëm tjetër joekzakt për krahasim?

Pyetja III: A është i mundur programimi i një aplikacioni që shfrytëzon algoritmin gjenetik dhe shërbimet e hartave për zgjidhjen e këtij problemi dhe që funksionon në pajisjet mobile?

Pyetja IV: Si do ta ndihmojë zbatimi i paketës softuerike (që është produkti përfundimtar i kësaj teze doktorale) zhvillimin e turizmit në Kosovë?

Pyetja V: A mund të jetë kjo paketë softuerike e zbatueshme në të gjitha trojet shqiptare dhe më gjerë?

1.3. Objektivat e studimit

- Programimi i algoritmit gjenetik dhe ofrimi i tij si ueb shërbim RESTful me Java.
- Konsumimi i këtij ueb shërbimi RESTful (algoritmi gjenetik) nga pajisjet mobile me sistem operativ Android.
- Integrimi i ueb shërbimeve të Google-it në Android, respektivisht hartat e Google-it, që mund t'i shoqërohen zgjidhjes së problemit biznesmeni udhëtar.
- Programimi i simulatorëve për testimin e algoritmit gjenetik dhe atij simulated annealing gjatë zgjidhjes së problemit biznesmeni udhëtar.
- Krahasimi i rezultateve të fituara nga simulatorët, për të nxjerrë në pah se cili nga këto dy algoritme optimizon rrugën më tepër dhe ka kohën e ekzekutimit më të shkurtër.
- Programimi i një aplikacioni për Smart-Phone, i cili shfrytëzon objektivat e cekur më lart, për t'i mundësuar një vizitori (udhëtari) gjetjen e rrugës më të shkurtër ose më pak të kushtueshme drejt vendeve historike, bukurive natyrore etj. ku e gjithë kjo ndodh grafikisht përmes hartave të Google-it.

1.4. Hipoteza

Zbatimi i algoritmit gjenetik, bazuar në implementimin e teknologjive të reja dhe pajisjeve mobile, do të ndikojë dukshëm në optimizimin dhe vizualizimin e rrugës te problemi biznesmeni udhëtar.

Programimi i një pakete të tillë softuerike, do të nxisë dukshëm turizmin në trevat shqiptare; veçanërisht turizmin në Kosovë.

1.5. Motivimi për të studiuar problematikën e zgjedhur

Në zgjedhjen e kësaj teme jam nxitur meqë problemi biznesmeni udhëtar ka mbajtur gjallë interesat e shkencëtarëve të shkencave kompjuterike dhe matematikanët, sepse, edhe pas rreth gjysmë dekade të hulumtimit, problemi nuk është zgjidhur plotësisht.

TSP gjen aplikim në shumë probleme të jetës reale, si tek: industri në organizimin e prodhimit, në gjetjen më optimale për radhitjen e operacioneve dhe detyrave; planifikimi i rrugëtimit, transportit dhe logjistikës.

Jam nxitur, po ashtu, nga dëshira për të programuar një aplikacion (apo një paketë softuerike) që të kontribuonte për turizmin në trevat shqiptare, me qëllimin që vizitorët e huaj ta kenë më lehtë gjetjen e vendeve me vlerë historike, ose me bukuri natyrore etj.

1.6. Mundësitë dhe kufizimet

Para se algoritmi gjenetik ta fillojë punën e vet, për të optimizuar gjatësinë e rrugës, që fillim duhet të dihen distancat në mes rrugëve që të çojnë te vendet që do të vizitohen.

Në realitet rrugët që çojnë nga pika A në pikën B mund të jenë një apo dydrejtimëshe. Për shkak të thjeshtimit të problemit do të supozoj se të gjitha vendet që të çojnë drejt vendeve, që do të vizitohen, janë dydrejtimëshe.

Google ofron ueb shërbime në formë hartash, Google Distance Matrix API dhe Google Geocoding Api. Distance Matrix ofron distancën në mes dy vendeve, ndërsa geocoding shndërron gjerësinë dhe gjatësinë gjeografike (latitude dhe longitude) në adresë dhe anasjelltas. Këto rezultate ofrohen në dy formate: JSON dhe XML. Google Distance Matrix API dhe Google Geocoding API nuk ofrohen për vendet që nuk janë anëtare të OKB-së, siç është p. sh.: Kosova. Sikur të përdorej vetëm ky ueb shërbim, atëherë softueri përfundimtar, që është produkti final i punës praktike të kësaj teze doktorale, do të ishte i kufizuar dhe do të funksiononte vetëm për vendet anëtare të OKB-së. Pra, me një fjalë do të ishte i kufizuar. Në mënyrë që sistemi i programuar të mos ketë kufizime të tilla, është i nevojshëm edhe implementimi i formulës së Haversinës për llogaritjen e distancës mes dy vendeve, duke u bazuar në gjatësinë dhe gjerësinë gjeografike të tyre.

Pra, si përfundim duhet implementuar një algoritëm gjenetik me dy funksione të ndryshme për llogaritjen e gjatësisë së turit, ku njëri implementon ueb shërbimin e Google-it: Google Distance Matrix për vendet që janë anëtare të OKB-së, ndërsa tjetri implementon formulën e Haversinës për vende që nuk janë anëtare të OKB-së. Duhet pasur parasysh që formula e Haversinës gabon në llogaritjen e gjatësisë së turit, meqë ajo nuk merr parasysh rrugën reale që lidh dy vende në terren. Ajo supozon që gjithnjë ka një rrugë të drejtpërdrejtë që lidh pikën A dhe pikën B, ndërkohë që në terren ekziston një rrugë e tërthortë, që i lidh këto dy pika (përmes një pike tjetër C) dhe që është më e gjatë sesa ajo e supozuara. Pra, në disa

raste mund të marrim si rezultat një gjatësi jo të saktë të turit dhe si rrjedhojë algoritmi mund të marrë vendime të gabuara në lidhje me renditjen e vendeve për vizitë.

1.7. Rëndësia e studimit dhe kontributi i tij

Përveç rëndësisë akademike ka dhe një rëndësi tjetër se për herë të parë bëhet kombinimi i algoritmit gjenetik me ueb shërbimet të Google-it dhe programimi i një aplikacioni që vepron në pajisjet me sistem operativ Android, për zgjidhjen e TSP-s, ky studim mund të ketë rëndësi kombëtare sepse: ky aplikacion mund të kontribuojë në turizmin në trevat shqiptare duke ua lehtësuar vizitorëve të huaj gjetjen më të lehtë të vendeve me vlerë historike, dhe të atyre me bukuri natyrore etj. Ata përmes aplikacionit do të zgjedhin vendet që dëshirojnë t'i vizitojnë, ndërsa aplikacioni ua vizaton në smart-phone, në hartën e Google-it, rrugën më të shkurtër dhe më pak të kushtueshme për në ato vende, dhe po kështu mund të kthehen prapë te vendi i nisjes.

1.8. Metodat dhe metodologjia e kërkimit

Fillimisht kam bërë përmbledhjen e literaturës duke hulumtuar përmbajtjen e bazës së të dhënave bibliografike, duke përdorur strategji të ndjeshme gjatë kërkimit. Pra, duke kërkuar në bazë të fjalëve kyçe. Gjithashtu, kam përdorur një referencë të një burimi të dhënë për lokalizimin e burimeve plotësuese. Pra, për sa i përket pjesës teorike kam analizuar publikimet e ndryshme shkencore, disertacione të ndryshme, të cilat do t'i shpjegoj në kapitullin shqyrtimi i literaturës së radhitur në bazë të autorit. Kjo metodë më ka ndihmuar gjatë shkrimit të doktoratës për të shmangur plagjiaturën e paqëllimtë.

Ndërsa për sa i përket pjesës praktike ose programimit të projektit doktoral, kam përdorur literaturën profesionale të programimit, ku vlen të theksohen Design Patterns (mënyra ose shabllone të ndryshme të programimit të propozuara nga ekspertë të ndryshëm) për probleme të ndryshme, tutorial të ndryshme në lidhje me RESTful të JAVA-s si dhe Google Maps Android API v2 (dokumentacioni online i Google-it për android në lidhje me përdorimin e hartave). Natyrisht më ka shërbyer provoja shumëvjeçare si programues.

1.9. Struktura e punimit doktoral

Kapitulli I: Hyrja

Është konceptuar si hyrje e studimit. Pasi bëhet shtrimi i problemit, jepen qëllimi i punimit dhe objektivat e tij; renditen pyetjet kërkimore dhe hipoteza bazë që do të vërtetohet. Më pas flitet për rëndësinë e studimit dhe përfitimet prej tij, për literaturën e shfrytëzuar; për metodologjinë e punës dhe për metodat e përdorura për këtë hulumtim. Në fund jepet struktura e punimit.

Kapitulli II: Problemi biznesmeni udhëtar dhe hulumtime të ngjashme

Origjina, klasifikimi, shtrimi i problemit. Këtu përshihet definimi matematik i problemit, aplikimet e TSP-s dhe problemet e ngjashme me të, si dhe algoritmet ekzakte dhe joekzakte për zgjidhjen e këtij problem.

Hulumtimi dhe leximi se çfarë kanë bërë të tjerët në lidhje me problemin biznesmeni udhëtar (Travelling Salesman Problem), përfshi këtu punime diplomash, papers të ndryshme etj, thënë shkurt radhitja në bazë të autorëve ashtu si i janë qasur problemit dhe çfarë kanë ardhur në përfundim në lidhje me të njëjtin problem.

Kapitulli III: Arkitektura e orientuar drejt shërbimeve (SOA)

Hyrje në arkitekturën e orientuar drejt shërbimeve, komponentët arkitekturore të SOA-s, përparësitë dhe sfidat e SOA-s. Ueb shërbimet si pjesë e pandashme e SOA-s, shtresat e ueb shërbimeve, arkitektura dhe llojet e tyre: RESTful dhe SOAP.

Pasqyra e përgjithshme në lidhje me ueb shërbimet e Google-it, kërkesa dhe përgjigjja nga serveri i Google-it, përfshi dhe parametrat obligativë dhe opcionalë. Këtu dallohen këto ueb shërbime të Google-it:

- Google Maps Distance Matrix API
- Google Maps Directions API
- Google Maps Geocoding API
- Google Maps Geolocation API
- Google Places API Web Service
- Google Maps API

Në këtë kapitull, po ashtu, bëhet një hyrje në kompjuterizimin në re, si teknologji dhe infrastrukturë në të cilën ofrohen gjitha ueb shërbimet e sodit pa dallim, flitet për përparësitë e tij dhe cilat janë modelet e shërbimit në kompjuterizimin në Re (cloud

computing): softueri si shërbim (SaaS), platforma si shërbim (PaaS) dhe infrastruktura si shërbim (IaaS).

Modelet e shpërndarjes së shërbimeve të Kompjuterizimit në Re: Private Cloud, Public Cloud dhe Hybrid Cloud. Kompjuterizimit në Re dhe virtualizimi, barrierat në retë kompjuterike si dhe rreziqet në retë kompjuterike.

Rëndësi e veçantë i është kushtuar edhe Mobile Cloud Computing (MCC) si një koncept i pranuar gjerësisht, që ka për qëllim të përdorë teknikat e cloud computing-ut për ruajtjen dhe përpunimin e të dhënave për pajisjet mobile, duke bërë të mundur uljen e kufizimeve të tyre.

Kapitulli IV: Programimi i sistemit të propozuar

Hyrje në algoritmin gjenetik dhe programimi i tij. Programimi i tij në JAVA në mënyrë që të ofrohet si ueb shërbim. Diskutimi në lidhje me parametrat që ndikojnë punën e tij si dhe kushtet që duhet të plotësohen që algoritmi të pushojë së ekzekutuari. Po ashtu në këtë kapitull rëndësi e njëjtë i kushtohet edhe programimit të aplikacionit për Android, duke përfshi këtu edhe integrimin e ueb shërbimeve të Google-it.

Hyrje në sistemin operativ Android si sistem operativ me kod burimor të hapur, bazuar në Linux, karakteristikat dhe versionet e tij.

Programimi i Manifestit të Android-it, skedar ky ku deklarohen dhe definohen të gjitha aktivitetet e aplikacionit, themes, orientimi i ekranit, premissions etj. Çdo send në këtë

skedar programohet në XML. Programimi i GUI-s së parapëlqimeve përmes XML-it si dhe funksionalizimi i tij me gjuhen programuese JAVA.

Po ashtu përmes screenshots sqarohet që si përdoruesi shërbehet me këtë aplikacion, ndërsa përmes shkëputjes së pjesëve kyçe të kodit në JAVA janë sqaruar si janë programuar events dhe sendet e tjera.

Kapitulli V: Analizë, testim dhe interpretim rezultatesh

Programimi i simulatorëve për testimin e algoritmit gjenetik dhe algoritmit simulated annealing, përmes të cilëve testohet dhe demonstrohet grafikisht puna e këtyre dy algoritmeve gjatë zgjidhjes së problemit TSP në rrafshin (x, y). Krahasimi i rezultateve dhe interpretimi i tyre, ku del në pah se algoritmi simulated annealing e optimizon rrugën më tepër, por është edhe më i shpejtë. Këtu bëhet edhe testimi tërësor i sistemit client-server, testimi i të gjithë punës praktike të doktoratës sime. Këtu dokumentohet me pamje (sreen shuts) nga aplikacioni për telefon si dhe kohëzgjatja e secilit proces veç e veç, por po ashtu dhe e të gjithë sistemit në përgjithësi.

Kapitulli VI: Përfundime dhe rekomandime

Ngjashmëritë dhe dallimet në mes algoritmit gjenetik dhe atij simulated annealing gjatë zgjidhjes së problemit TSP si dhe në përgjithësi. Konkluzione në lidhje me algoritmin gjenetik në përgjithësi. Puna në të ardhmen është po ashtu njëra nga pikat që diskutohet në këtë kapitull.

KAPITULLI II: PROBLEMI BIZNESMENI UDHËTAR DHE HULUMTIME TË NGJASHME

2.1. Historiku

Problemi biznesmeni udhëtar (TSP) është studiuar gjerësisht në shkencat kompjuterike. Ka shumë publikime mbi TSP-në, duke filluar që nga fund i viteve 1940. TSP ka mbajtur gjallë interesat e shkencëtarëve të shkencave kompjuterike dhe matematikanët, sepse, edhe pas rreth gjysmë dekade të hulumtimit, problemi nuk është zgjidhur plotësisht. TSP bën pjesë në një kategori të shquar të problemeve që janë të vështira për t'u zgjidhur. TSP mund të aplikohet për të zgjidhur shumë probleme praktike në jetën tonë të përditshme. Kështu, një zgjidhje për TSP-në do të jetë shumë i dobishëm. Kur ne flasim për një zgjidhje të TSP-së, ne flasim për një zgjidhje në kohë polinomiale të TSP-së së përgjithshme. (The Traveling Salesman Problem: A Comprehensive Survey, 2006). Travelling Salesman Problem (TSP) përfaqëson një klasë të madhe të problemeve të njohura si probleme kombinatorike të optimizimit. Mes tyre, TSP është një nga më të rëndësishmet, meqë ky problem është shumë i lehtë për t'u përshkruar, por shumë i vështirë për t'u zgjidhur.

Problemin e biznesmenit udhëtar e hasim çdo ditë në jetën e përditshme, në fusha apo lëmi të ndryshme të biznesit dhe të industrisë. Pra, gjetja e rrugës më të shkurtër për të vizituar partnerët e biznesit, vajtja dhe marrja e pajisjeve të prishura si dhe dërgimi i tyre për rregullim etj. (Stanec, 2011)

TSP mund të përdoret gjithkund, në shumë aktivitete njerëzore. Një nga më të rëndësishmit është organizimi i prodhimit, në gjetjen më optimale për radhitjen e operacioneve dhe

detyrave. Zbatim tjetër të rëndësishëm ky problem gjen te procesi i shpimit të vrimave te pllakat elektronike. (Mataija, Šegić, & Jozić, 2016)

TSP është aplikuar në shumë probleme të jetës reale, si planifikimi i rrugëtimit, transportit dhe logjistikës. Për këtë arsye, ky është një problem ideal testimi për të gjithë studiuesit ose hulumtuesit, sidomos për ata që punojnë në teknikat e optimizimit. Janë propozuar algoritme ekzakte për zgjidhjen e këtij problemi siç janë: programimi dinamik, branch-and-bound dhe branch-and-cut. Këto algoritme japin rezultate të suksesshme deri në dimensione të caktuara të këtij problemi. Te këto algoritme koha e përllogaritjes (ekzekutimit) rritet me rritjen e numrit të vendeve për vizitë dhe me rritjen e përmasave të problemit. Kjo është arsyeja pse bëhet i pamundur përdorimi i tyre, meqë koha e ekzekutimit të tyre nuk është brenda suazave të pranueshme. Në anën tjetër synohet të zbatohen metodat deduktive dhe metadeduktive për zgjidhjen e këtij problemi, për shkak të kohës së tyre të shkurtër të përllogaritjes, megjithatë nuk garantojnë rezultat optimal. Këto metoda janë shumë të preferuara nga hulumtuesit e kësaj fushe. (Ilhan, 2016)

2.2. Origjina e problemit

Problemi i biznesmenit udhëtar (Travelling Salesman Problem-TSP) është studiuar në shekullin e 18-të nga një matematikan irlandez, i quajtur Sir William Rowam Hamilton, dhe nga matematikani britanik, i quajtur Thomas Pennington Kirkman. (Biggs, Lloyd, & Wilson, 1986)

Një diskutim i hollësishëm për punën e Hamilton & Kirkman mund të shihet nga libri me titull teoria e grafeve. (Biggs, Lloyd, & Wilson, 1986)

Besohet se forma e përgjithshme e TSP-së mund të jetë studiuar për herë të parë nga Karl Menge në Vjenë dhe në Harvard. (Biggs, Lloyd, & Wilson, 1986)

2.3. Definimi i problemit

Një shitës planifikon një udhëtim përmes të cilit dëshiron t'i vizitojë klientët e tij dhe të kthehet prapë në pikën e nisjes. Gjatë rrugës vajtje-ardhje klienti i njëjtë nuk duhet të vizitohet më tepër se një herë, por edhe asnjë klient nuk duhet lënë pa u vizituar, ndërsa rruga duhet të jetë sa më e shkurtër dhe sa më pak e kushtueshme.

2.4. Klasifikimi i problemit

Në përgjithësi, problemi i biznesmenit udhëtar klasifikohet si: simetrik (sTSP), dhe asimetrik ose josimetrik (aTSP). (Matai, Singh, & Mittal, 2010)

Në sTSP distanca në mes të vendeve për t'u vizituar është e njëjtë në çdo drejtim të kundërt, duke formuar kështu një graf të paorientuar. Kjo simetri përgjysmon numrin e zgjidhjeve të mundshme.

2.5. Konceptet themelore të teorisë së grafeve

Te problemi aTSP nuk ekzistojnë rrugë dykahore (dydrejtimëshe), ose distanca mes tyre mund të jetë e ndryshme, duke formuar kështu një graf të orientuar. (Matai, Singh, & Mittal, 2010)

Një graf G=(V, E) përbëhet nga një bashkësi joboshe, por të fundme nyjesh V = {1, 2, 3, ..., n} dhe nga një bashkësi e fundme brinjësh E = {(i, j)| $\exists$ një lidhje direkt nga nyja i $\in$ V te nyja j $\in$ V}. Nëse mbi brinjë qëndrojnë (janë dhënë) vlera c(i,j) $\in \mathbb{R}^+$, atëherë është fjala

për graf të peshuar ose të vlerësuar (anglisht: weighted). Vlerat mbi brinjët e grafit simbolizojnë distancën, kohën e udhëtimit ose shpenzimet e udhëtimit mes dy nyjeve i dhe j. Vlerat mbi brinjët e grafit shpesh quhen edhe pesha. Vlerat e brinjëve zakonisht ruhen në të ashtuquajturën matricë e kostos, e cila ka formën e mëposhtme:

C = $(c_{i,j})$ ku $c_{i,j} = \begin{cases} c_{i,j}, & nërastse (i,j) \in E,\ i \neq j \\ \infty, & përndryshe \end{cases}$

Dallojmë dy lloje grafesh: ato me brinjë me drejtim të orientuar dhe me drejtim te paorientuar.

Te grafet me drejtim të orientuar, çdo brinjë ka një nyje fillestare dhe një përfundimtare, kështu që në mes brinjës (i, j) dhe brinjës së kundërt (j, i) ka një dallim në kahje (drejtim). Për brinjën (i, j) të një grafi, 'i' është paraardhësi i 'j' dhe 'j' është pasardhësi i 'i'. P(i) është bashkësia e paraardhësve të 'i'-së, ndërsa S(i) është bashkësia e pasardhësve të 'i'-së. Në këtë rast nyjet i dhe j janë fqinjë. Grada δ(i) tregon numrin e fqinjëve të nyjes 'i'.

Një rrugë nga u $\in$ V deri v $\in$ V është një varg i brinjëve ((u, ω_1), (ω_1, ω_2), ... (ω_{m-1}, v)) $\in E^m$ ku m $\geq$ 1. Nyja 'v' konsiderohet e arritshme nga 'u'. Bashkësia R(i) = R(i) \ {i} paraqet bashkësinë e nyjeve të arritshme nga 'i', mirëpo që janë të ndryshme nga 'i'. Një rreth i grafit formohet nëse nyja fillestare dhe përfundimtare e një rruge të grafit janë identike, prej nga rrjedh që u = j.

Një graf quhet i lidhur nëse për çdo palë nyjesh u,v $\in$ V ekziston një rrugë mes tyre. Mund të formohen cikle të Hamiltonit duke ndërtuar sekuenca brinjësh, të cilat përmbajnë çdo nyje të grafit saktësisht vetëm njëherë. Një linje e mbyllur e Eulerit është një sekuencë brinjësh në të cilën çdo brinjë e grafit është e përmbajtur vetëm një herë.

Në vijim do të kufizohem vetëm te grafet e lidhura dhe simetrike të cilat nuk përmbajnë brinjë të shumëfishta. (Kämpf, 2006)

2.6. Formulimi matematik i problemit

Formulimi përmes grafeve

TSP mund të definohet përmes një grafi të plotë dhe të paorientuar $G = (V, E)$ nëse problemi është simetrik, ndërsa përmes grafit të orientuar $G = (V,A)$ definohet problemi asimetrik (josimetrik). Le të jetë bashkësia V = {1, . . . , n} bashkësi nyjesh të grafit, bashkësia $E = \{(i, j) : i, j \in V, i < j\}$ bashkësi brinjësh të grafit dhe $A = \{(i, j) : i, j \in V, i \neq j\}$ bashkësia e brinjëve të orientuara (me kahje të orientuar). Matrica $C = (c_{ij})$ është e definuar në E ose A dhe quhet matrica kosto (ku ruhen distancat mes nyjeve). Matrica C kënaq jo barazinë $c_{ij} \leq c_{ik} + c_{kj}$ për çdo i, j, k. Në veçanti, ky është rasti ku çdo nyje është pikë në rrafshin XY, dhe $c_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$ quhet distanca e euklidit. Ky jobarazim vlen nëse c_{ij} është gjatësia e shtegut më të shkurtër në mes i-së dhe j-së në grafin G. (Matai, Singh, & Mittal, 2010)

Thënë shkurt te problemi i biznesmenit udhëtar kërkohet një rreth i Hamiltonit me gjatësi minimale, në të cilin përmbahen të gjitha nyjet e grafit nga një herë.

Formulimi si programim linear

Sipas përkufizimit të TSP-së, përshkrimi i saj matematikor është si vijon:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ për } i = 1 \ldots n \qquad (2)$$

$\sum_{i=1}^{n} x_{ij} = 1$ për j = 1 … n (3)

$x_{ij} \in \{0, 1\}$ (4)

me $x_{i,j} = \begin{cases} 1, & nërast \exists\ lidhje\ direkte\ prej\ i\ një \\ 0, & përndryshe \end{cases}$ (5)

Gjatë kërkimit për të gjetur zgjidhjen optimale është i domosdoshëm shtimi i një kushti shtesë, për të shmangur kështu turet e shkurta (ku nuk përfshihen të gjitha nyjet e grafit):

$\sum_{\{i,j\} \subseteq S} x_{ij} \leq |S| - 1 \quad \forall S \subset V, |V| \geq 2$ (6) (Kämpf, 2006)

2.7. Aplikimet e TSP-së dhe problemet e ngjashme me të

Kabllimi në kompjuter (Computer wiring)

Këtë problem e hasim shpesh gjatë projektimit të ndërfaqeve (Interfaces) kompjuterike. Një ndërfaqe përbëhet nga një numër modulesh, dhe në secilin nga këto module gjenden pin-a të ndryshëm. Pozicioni i çdo moduli është i paracaktuar më parë. Një nënbashkësi e pinë-ve duhet të ndërlidhen mes veti me tela. Problemi duhet trajtuar nga këndvështrimi i mundësive për ndryshim dhe për korrigjim të pinë-ve në të ardhmen, dhe po ashtu duhet marrë parasysh madhësia e vogël e PIN-it si dhe çdo PIN-i maksimumi i duhen bashkangjitur dy tela. Në mënyrë që të shmanget kalimi i telit mbi tel, që moduli të jetë sa më i lehtë dhe kabllimi të jetë sa më i pastër (në kuptimin që të mos jetë i mbingarkuar), gjatësia e telit duhet të minimizohet (të jetë sa më e shkurtër e mundshme). (J. K. Lenstra, 1975)

Rrugëtimi i automjeteve-kamionëve (Vehicle Routing)

Supozojmë që në një qytet gjendet një numër i caktuar kutish postare, të cilat brenda një periudhe të caktuar kohore duhet të zbrazen çdo ditë, le të themi brenda një ore. Problemi është për të gjetur numrin minimal të kamionëve për të bërë mbledhjen ose koleksionimin brenda kohës së paraparë me këtë numër kamionësh (pa numër shtesë kamionësh). Ja edhe një shembull që na duhet: ekziston një numër i caktuar klientësh (n) që kërkojnë sasi të caktuar mallrash dhe një furnitor që duhet t'i furnizojë ata ose që t'i përmbushë këto kërkesa përmes një numri të caktuar kamionësh. Sfida e këtij problemi është caktimi i kamionëve për te klientët sipas një plani (orari) të caktuar furnizimi, por në mënyrë të tillë që ngarkesa e lejuar për kamion të mos tejkalohet dhe distanca që do të përshkojë çdo kamion të jetë minimalja më e e mundshme. (J. K. Lenstra, 1975)

Rrugëtimi i autobusëve shkollorë

Plani i rrugëtimit të autobusëve shkollorë është investiguar si variant e problemit mTSP me disa kufizime anësore. Qëllimi i këtij planifikimi kohor të ngarkesës së autobusëve është gjetja e një mostre, kështu që numri i rrugëve për secilin autobus të jetë minimal, gjatësia e rrugës të të gjithë autobusëve të jetë sa më e vogël dhe që asnjë autobus të mos mbingarkohet me nxënës shkolle (të lejohen vetëm aq sa është numri i ulëseve) si dhe të respektohet plani kohor i frekuentimit të çdo rruge. (Angel, Caudle, & Noonan, 1972)

Design of global navigation satellite system surveying networks (GNSS)

Një GNSS është një sistem satelitor, me bazë në hapësirë, i cili siguron mbulim për të gjitha vendet në mbarë botën dhe është mjaft i rëndësishëm në aplikime të jetës reale, si në paralajmërimin e hershëm dhe në menaxhimin e fatkeqësive, të mjedisit dhe monitorimin e

bujqësisë, etj. Qëllimi i vëzhgimit është për të përcaktuar pozicionet gjeografike të pikave të panjohura mbi tokë duke përdorur pajisje satelitore. Këto pika, në të cilat janë vendosur marrës, koordinohen nga një seri njësish për vëzhgim. Kur ka marrës të shumtë apo periudha të shumta të punës, problemi i gjetjes së rendit më të mirë të seancave për marrës mund të formulohet si një mTSP. (Saleh & Chelouah, 2004)

Procesi i shpimit te pllakat elektronike

Për ta lidhur një përçues në njërën anë me një përçues në shtresën tjetër të pllakës elektronike, apo për t'i pozicionuar PIN-ët e elementeve të zhytura elektronike, duhet që në pllakë të hapen vrima. Vrimat që shpohen duhet të jenë të madhësive të ndryshme dhe kjo si rrjedhojë e madhësive të ndryshme të elementeve që duhen vendosur në pllakë. Për t'i shpuar dy vrima me diametër të ndryshëm njëra pas tjetrës, duhet që koka e makinës së shpimit të lëvizë deri te kutia e veglave për ta ndryshuar pajisjen e shpimit. Kjo është e kushtueshme në aspektin kohor. Diçka duhet të vendoset: ose në fillim duhet të shpohen të gjitha vrimat me diametër të njëjtë dhe mandej të ndërrohet pajisja për shpim dhe kështu të vazhdohet më shpimin e vrimave të tjera me diametër tjetër e kështu me radhë të vazhdojë procesi. Pra, problemi i shpimit mund të shihet si një seri e TSP-së, ku për çdo diametër të vrimës (që simbolizon vendin për vizitë), që paraqet pozicionin fillestar, vendosen të gjitha vrimat me të njëjtin diametër dhe që shpohen me të njëjtën pajisje shpimi. E 'distance' në mes të dy qyteteve është dhënë nga koha që duhet për të lëvizur kokën e shpimit nga një pozicion në tjetrin. Qëllimi është për të minimizuar kohën e udhëtimit për kokë makinë. (Grötschel, Jünger, & Reinelt, 1991)

2.8. Metodat numerike (algoritmet) për zgjidhjen e problemit TSP

Algoritmet ekzakte

Ekzistojnë dy grupe algoritmesh ekzakte për zgjidhjen e problemit TSP. Një nga këto është zgjidhja e problemit TSP, e formuluar si Programim linear dhe që shfrytëzon metodat: Branch and Bound dhe Branch and Cut dhe Brute Force. Një grup tjetër i vogël përdor Programimin dinamik. Për të dyja grupet karakteristika kryesore është garancia e gjetjes së zgjidhjeve optimale duke marrë në konsideratë kohën e ekzekutimit dhe të hapësirës për ruajtje. (Ch. Chauhan & Pathak, 2012)

Branch and Bound

Procesi i degëzimit

Principi i punës së këtij algoritmi është degëzimi dhe kufizimi. Gjatë procesit të degëzimit problemi fillestar ndahet në dy ose më shumë nënprobleme dhe paraqitet si strukturë e të dhënave pemë, me ç'rast bëhet thjeshtësimi i problemit të dhënë. Gjatë këtij procesi rekursiv përdoren tri metoda më të njohura: (Jakob I. Bernoulli)

1) Kërkimi në thellësi

Nga nënproblemet ende të pazgjedhura, përzgjidhet i pari ai problem i cili në strukturën e trungut ka hyrë si nyje e fundit. Në këtë mënyrë, shumë shpejt përpunohen të gjitha nyjet në thellësinë e trungut, me ç'rast përfitohet një zgjidhje, mirëpo në lidhje me cilësinë e kësaj zgjidhjeje nuk mund të thuhet asgjë. (Jakob I. Bernoulli)

2) Kërkimi në gjerësi

Nga nënproblemet ende të pazgjedhura, përzgjidhet i pari ai problem i cili në strukturën e trungut ka hyrë si nyje e parë. Kështu në mënyrë rekursive bëhet përpunimi i nyjeve të

trungut të të njëjtit nivel pa u futur në thellësinë e trungut. Gjatë kësaj metode në fund pothuajse një zgjidhje e lejuar, mirëpo kjo zgjidhje ka tendencë të jetë një zgjidhje e mirë, pra të përzgjidhet në procesin e mëtejshëm në zgjidhjen e problemit fillestar (të përgjithshëm). (Jakob I. Bernoulli)

3) Kërkimi i më së mirës

Nga nënproblemet ende të pazgjedhura, përzgjidhet i pari ai problem që shfaq ose tregon kufirin më të ulët të mundshëm. Përmes këtij rregulli të përzgjedhjes cilësore, bëhet i mundur përfitimi i një zgjidhjeje cilësore. (Jakob I. Bernoulli)

Procesi i kufizimit

Gjatë këtij procesi bëhet prerja e degëve të caktuara të trungut të cilat nuk duhet të merren në konsideratë gjatë llogaritjeve të mëtejshme, kështu që bëhet shkurtimi i kohës së ekzekutimit të algoritmit. Këtë e arrin algoritmi duke bërë llogaritjen dhe krahasimin e dy rangjeve (kufijve) të dy palë nyjeve në trung. Rruga nga rrënja e trungut e deri te nyja ku është pozicionuar nënproblemi përbën kufirin e poshtëm (rangun) e problemit. (Jakob I. Bernoulli)

Në punimin e tyre autorët (Babar, Jadhav, Jagtap, & Joshi, 2014) thonë se strategjia Branch and Bound ndan problemin që duam ta zgjidhim në një numër të nënproblemeve dhe më pas rezultatin e këtyre nënproblemeve e kombinon për të marrë rezultatin përfundimtar. Ky është një sistem për zgjidhjen e një vargu nënproblemesh, ku secili prej tyre mund të ketë zgjidhje të shumta të mundshme, dhe se zgjidhja e përzgjedhur e një nënproblemi mund të ndikojë në zgjidhjet e nënproblemeve të mëvonshme.

Autori: Chaitanya Pothineni në punimin e tij (Pothineni, 2013) paraqet një pseudo-code sesi funksionon në përgjithësi metoda Branch and Bound:

(i) Le të jetë S nënbashkësia e zgjidhjeve të një problemi që duam ta zgjidhim;

(ii) L(S) -> kufiri i poshtëm i kostos së një zgjidhjeje nga S;

(iii) Le të jetë C kostoja e zgjidhjes më të mirë e gjendur deri me tani;

(iv) Në rast se C $\leq$ L(S), nuk është e nevojshme që S të përpunohet më tutje, meqë nuk përmban zgjidhje më të mira;

(v) Në rast se C >L(S), atëherë përpunimi i mëtejshëm i S-së është i nevojshëm, meqë mund të përmbajë zgjidhje më të mira sesa ato të gjetura deri tani.

Pseudo-code i algoritmit Branch and Bound

Hapi I: Zgjedhja e nyjes fillestare në trung (apo graf).

Hapi II: Vendosja e kufirit në një vlerë shumë të madhe, le të themi baras me pakufi.

Hapi III: Përzgjedhja e brinjës më të shkurtër në mes nyjes aktuale dhe një nyje tjetër të pavizituar, dhe shtimi i gjatësisë së saj në gjatësinë aktuale, dhe përsëritja e këtij hapi përderisa gjatësia aktuale është më e vogël sesa vlera e vendosur kufitare.

Hapi IV: Nëse gjatësia aktuale është më e vogël sesa vlera kufitare, atëherë puna mbaroi.

Hapi V: Shtoni gjatësinë dhe vlera kufitare do të jetë e barabartë me gjatësinë aktuale.

Hapi VI: Hapi V duhet përsëritur derisa të mos jenë vizituar (shfrytëzuar) të gjitha brinjët e grafit. (Saiyed, 2012)

Në punimin e tyre (Mataija, Šegić, & Jozić, 2016) autorët treguan rrugën sesi të zgjidhet problemi TSP duke kombinuar tabela të dhënash dhe algoritmin branch-and-bound. Kjo metodë dha zgjidhjen e duhur dhe vërtetoi të jetë e suksesshme në zgjidhjen e

këtij problemi, deri në 60 vende që janë vizituar. Ata përdorën metodën për zgjidhjen e problemit të njohur, dërgimin e paketave tek adresa të caktuara ku distanca mes tyre dihet që nga fillimi. Përmes kësaj metode bëhet përllogaritja optimale e turit për dërgim paketash, si dhe jepet gjatësia e tij në metra.

Hulumtimi (Mataija, Šegić, & Jozić, 2016) ka treguar se metoda për të llogaritur rrugët optimale, e përdorur në këtë mënyrë dhe pa përdorimin e paketave softuerike shtesë, mund të jetë efikase.

Në kontekstin e tregut konkurrues është e rëndësishme për të racionalizuar çdo segment të biznesit duke përdorur metodat matematikore të tilla si kjo. Racionalizimi nga llogaritja e rrugëve optimale është shumë i thjeshtë për t'u përdorur dhe kjo redukton kostot e biznesit në mënyrë të konsiderueshme. Metoda branch-and-bound është e përshtatshme për të punuar me aplikimet kompjuterike dhe është konsideruar si një nga mënyrat e mundshme për të inkurajuar eksplorimin e mëtejshëm të këtij problemi. (Mataija, Šegić, & Jozić, 2016)

Branch and Cut

Metoda branch and cut zgjidh programin linear pa përkufizuar problemin në numra të plotë duke përdorur algoritmin e rregullt simplex. Nëse arrihet një zgjidhje optimale, dhe kjo ka vlerë optimale nëse nuk është numër i plotë për një variabël, por që është supozuar të ishte e tillë, atëherë përdoret algoritmi Cutting Plane për gjetjen e kufizimeve lineare shtesë. (Ch. Chauhan & Pathak, 2012)

Ideja themelore

1) Gjej zgjidhjet e mundshme përmes 0/1-vektorëve.
2) Përshkrimi sa më i mirë i mundshëm i sipërfaqes konvekse që formojnë këta vektor, përmes pabarazimeve lineare.
3) Gjetja e algoritmeve sa më të mirë që të garantojnë që nuk ekzistojnë pika të cilat i lëndojnë këto jobarazime.
4) Ndarja (prerja) (Ch. Chauhan & Pathak, 2012)

Programimi dinamik

Programimi dinamik është një qasje për optimizim që transformon një problem kompleks në një varg problemesh të thjeshta. Karakteristikë e tij thelbësore është natyra shumëshkallëshe e procedurës së optimizimit. Më shumë se teknikat për optimizim, programimi dinamik ofron një framework të përgjithshëm për analizimin e shumë llojeve të problemeve. Përmes këtij framework-i dhe falë shumëllojshmërisë së teknikave për optimizim, mund të zgjidhen aspekte të veçanta të një formulimi më të përgjithshme. Zakonisht kreativiteti është i nevojshëm më parë sesa të mund të njohin që një problem i veçantë mund të hidhet në mënyrë efektive si një program dinamik; dhe shpesh njohuritë delikate janë të nevojshme për ristrukturimin dhe formulimin e problemeve të thjeshta, në mënyrë që ato të mund të zgjidhen në mënyrë efektive. (Bradley, Hax, & Magnanti, 1977)

Siç u cek më lart problemi origjinal ndahet në probleme të vogla. Problemet e vogla zgjidhen dhe zgjidhjet e tyre ruhen. Zgjidhjet e ruajtura ripërdoren disa herë nëse i njëjti rezultat i pjesshëm është i nevojshëm më shumë se një herë në zgjidhjen e problemit kryesor. Programimi dinamik bazohet shpesh në një formulë rekursive për zgjidhjen e problemeve të mëdha falë zgjidhjeve të problemeve të vogla. (Petrakis)

Programimi dinamik në zgjidhje për të zgjidhur TSP-në

- Numërimi i qyteteve 1,2,…n;
- 1 është pika fillimit dhe e mbarimit;
- D (i,j): distance nga nyja i deri te nyja j;
- D(b,S): gjatësia e rrugës më të shkurtër për vizitën e të gjitha qyteteve duke ndjekur një renditje të caktuar dhe duke përfunduar te nyja 1;
- S është nën bashkësi e {1,2,…n} –{1,b};
- TSP ka nevojë për të llogaritur D (1, {2, ... n}) dhe rendin e nyjeve;
- Programimi dinamik llogarit dhe ruan D (b,S) për secilën b;
- TSP llogarit të gjitha D (b, S) duke filluar nga bashkësia boshe S dhe bën llogaritjet derisa S = {2,3, ...n};
- Supozojmë se ne fillojmë nga b = 1;
- D (b,φ) = d(b,1). (Petrakis)

Koha e ekzekutimit

Algoritmi standard ka kohën e ekzekutimit Ω((n-1)!).

Nëse të njëjtat rrugë të pjesshme janë përdorur shumë herë, atëherë Programimi dinamik mund të jetë më i shpejtë! (Dynamic Programming)

Brute force

Kur dikush mendon zgjidhjen e TSP-së, metoda e parë që mund të vijë në mendje është metoda brute-force. Algoritmi brute-force bën thjesht gjenerimin e të gjitha tureve të mundshme dhe bën gjithashtu llogaritjen e distancave (ose gjatësive) të tyre. Turi më i shkurtër është kështu turi optimal. Koha e ekzekutimit të këtij algoritmi është: O(n!) (Ch. Chauhan & Pathak, 2012)

Kur mendohet në lidhje me zgjidhjen e problemit TSP, metoda e parë që mund të vijë në mendje është një metodë e quajtur brute-force. Metoda brute-force është e thjeshtë, të gjeneruarit e gjitha tureve të mundshme dhe llogaritja e gjatësive të tyre. Turi më i shkurtër është kështu turi optimal. Për ta zgjidhur TSP-në duke përdorur metodën Brute-force duhet të ndërmerren hapat e mëposhtëm:

(i) llogaritja e numrit të përgjithshëm të tureve;

(ii) vizatimi dhe radhitja e të gjitha tureve të mundshme;

(iii) llogaritja e gjatësisë së secilit prej tyre;

(iv) përzgjedhja e turit më të shkurtër, kjo është zgjidhja ekzakte (optimale).

(Saiyed, 2012)

Algoritmet joekzakte

Këto algoritme ofrojnë zgjidhje potencialisht jooptimale, por zakonisht janë më të shpejtë dhe ndahen në algoritme të përafërta (Approximation Algorithms) dhe algoritme deduktive (Heuristic Algorithms). (Ch. Chauhan & Pathak, 2012)

Simulated Annealing

Është një metodë e njohur kërkimi meta-deduktive, që është përdorur me sukses në zgjidhjen e shumë problemeve kombinatorike të optimizimit. Termi simulated annealing është adoptuar nga shkrirja e metaleve, ku bëhet përpjekje për të minimizuar energjinë e sistemit duke përdorur ftohje të ngadaltë derisa atomet të arrijnë një gjendje të qëndrueshme. Teknika e ftohjes së ngadaltë i lejon vetvetiu atomet e metalit të formojnë një strukturë të rregullt kristalore me dendësi të lartë dhe energji të ulët. Temperatura fillestare dhe shkalla në të cilën temperatura është reduktuar quhet orari i shkrirjes. (Ch. Chauhan & Pathak, 2012)

Te ky algoritëm problemi i gjetjes së zgjidhjes optimale është i ngjashëm me një proces të reduktimit të energjisë së përgjithshme të sistemit me anë të kalimit nga një gjendje në tjetrën. Te TSP një gjendje është definuar si një tur, ose si një permutacion i qyteteve për t'u vizituar. Energjia e një gjendje të një metali është gjatësia totale e një turi. Në çdo përsëritje të algoritmit, një gjendje kandidate për tranzicion, e njohur gjithashtu si një fqinj i gjendjes aktuale, është gjeneruar nga shkëmbimi (apo ndërrimi në renditje) i një sërë qytetesh. (Yang F. , 2010)

Turi fqinj vlerësohet në bazë të një funksioni të pranimit që bën kalimin nga një gjendje në tjetrën, nëse turi fqinj ka një energji të ulët ose të zgjedhur me një probabilitet që varet nga dallimi mes energjive dhe një parametri global T (i quajtur temperatura). (Brownlee, 2012)

Simulated Annealing (Brownlee, 2012)

```
Input: ProblemSize, iterations_max, temp_max
Output: Sbest
Scurrent ← CreateInitialSolution(ProblemSize);
Sbest ← Scurrent;
for i = 1 to iterationsmax do
    Si ← CreateNeighborSolution(Scurrent);
    temp_curr ← CalculateTemperature(i, tempmax);
    if Cost(Si) ≤ Cost(Scurrent) then
        S_current ← Si;
        if Cost(Si) ≤ Cost(Sbest) then
            S_best ← Si;
        end
    else if EXP((Cost(S_current) − Cost(S_i)) / temp_curr) > Rand() then
        Scurrent ← Si;
    end
end
return Sbest;
```

Tabela 1: Pseudocode i Simulated Annealing

Algoritmet gjenetike

Principi i punës së tij mbështetet në rastësi dhe në teorinë e Evolucionit, respektivisht në përzgjedhjen natyrore ku më i forti mbijeton.

Thelbi i principit të punës së këtij algoritmi është ndryshimi i bashkësisë së zgjidhjeve të mundshme hap pas hapi. Gjatë ndryshimeve zgjidhjet shumë të mira ruhen,

ndërsa ato shumë të këqija, me gjasë shumë të madhe, hidhen poshtë dhe kështu në bazë të variacioneve dhe kombinacioneve krijohen zgjidhje të reja. Tek algoritmi gjenetik zgjidhjet e vetme trajtohen si individ, ndërsa bashkësia e zgjidhjeve si popullsi. Gjatë ekzekutimit të algoritmit nga epoka në epokë, popullsia fillestare ndryshohet. Këto ndryshime pasojnë nga rikombinimi dhe mutacioni i materialit gjenetik.

Objektivi i algoritmit gjenetik është gjetja e zgjidhjeve optimale për problemet, tek të cilat, gjetja e zgjidhjes ekzakte dështon për arsye të kompleksitetit kombinatorik.

Algoritmi gjenetik (Harbich, 2007)

gjenero popullatën fillestare P;
përderisa (kriteri i mbarimit nuk përmbushet) {
llogarit fitnes (f) për të gjithë individët nga P;
përzgjedh në bazë të funksionit për vlerësim (f);
zbato operacionet gjenetike (mutacionin dhe rikombinimin).
}

Tabela 2: Forma e përgjithshme e algoritmit gjenetik

Funksionet bazë të algoritmit gjenetik janë:

- Popullimi
- Përzgjedhja
- Rikombinimi (Crossover)
- Mutacioni
- Fitness (Vlerësimi)
- Zëvendësimi

Popullata (Popullimi)

Gjenerimi i rastësishëm i një numri të caktuar zgjidhjesh të mundshme, që janë nënbashkësi të të gjitha zgjidhjeve të mundshme të problemit. Mbi këto zgjidhje algoritmi gjenetik fillon punën e tij.

Përzgjedhja

Ky është hapi tek i cili bëhet përzgjedhja e prindërve për individët e gjeneratës së ardhshme. Kjo ndodh me zgjedhjen e rastësishme të individëve të gjeneratës aktuale. Për t'i ngjasuar sa më tepër përzgjedhjes natyrore, duhet që individët me fitness (funksion vlerësimi) më të mirë, të përdoren më shumë ose më shpesh si prindër, sesa individët me fitness më të keq. Kështu përzgjedhja i shtyn individët në drejtim të zgjidhjeve më të mira. (Grubel, 2003)

Autorët: Noraini dhe Geraghty në punimin e tyre (Noraini & Geraghty, 2011) thonë se algoritmi bazë gjenetik përbëhet nga dy procese. Procesi i parë është zgjedhja e individëve me vlerë për prodhimin e brezit të ardhshëm dhe procesi i dytë është manipulimi i individëve të përzgjedhur dhe me vlerë për të formuar brezin e ardhshëm përmes teknikave të bashkëdyzimit dhe mutacioneve. Mekanizmi i përzgjedhjes përcakton se cilët individë me vlerë përzgjidhen për riprodhim dhe sa pasardhës do të gjenerohen nga çdo individ i përzgjedhur. Parimi kryesor i strategjisë së përzgjedhjes është: sa më i mirë është një individ, aq më e lartë është mundësia e tij për të qenë prind. Në përgjithësi, teknikat e bashkëdyzimit dhe ndryshimit të rastësishëm eksplorojnë hapësirën e kërkimit, ndërsa zgjedhja zvogëlon zonën e kërkimit në kuadër të popullsisë.

Strategjitë e ndryshme të përzgjedhjes, që përdoren tek algoritmi gjenetik, ndikojnë dukshëm në performancën e tij. Autorët thonë se ky studim ka për qëllim të shqyrtojë punën e algoritmit gjenetik, duke përdorur strategji të ndryshme të përzgjedhjes në zgjidhjen e problemit biznesmeni udhëtar .

Metodat e përzgjedhjes të trajtuara në këtë studim janë:

Përzgjedhja proporcionale në bazë të vlerësimit: Çdo individi i përcaktohet proporcionalisht një gjasë, bazuar në funksionin për vlerësim, në mënyrë që të përzgjidhen te faza e rikombinimit. Gjasa e përzgjedhjes p_s të individit '*i*' te një popullatë me madhësi 'n' shprehet sipas formulës:

$$p_s(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)}$$

Çdo individ shumëzohet (kopjohet) falë funksionit të tij vlerësues, ndërsa numri i pasardhësve, që pritet të formohen nga ky shumëzim, për çdo individ llogaritet sipas formulës:

e(i) = $p_s(i) * n = \frac{f(i)}{\sum_{j=1}^{n} f(j)}$ *n

Pra, në bazë të formulës së mësipërme reduktohet (zvogëlohet) gabimi në përzgjedhje të rastësishme.

Përzgjedhja në bazë të rangut: Individët renditen në bazë të funksionit të vlerësimit (Fitnes) dhe për përzgjedhje përdoret vetëm pozicioni brenda kësaj radhitjeje. Kjo lloj përzgjedhjeje është e përshtatshme për popullata tek të cilat shpërndarja e vlerave për vlerësim te 'përzgjedhja proporcionale në bazë të vlerësimit' është problematike. (Grubel, 2003)

Përzgjedhja Tournament: Prindërit përfitohen duke zgjedhur në mënyrë të rastësishme vetëm ata me vlerësim më të mirë. Ky variant mund të programohet në mënyrë efiçente dhe është i përshtatshëm për popullata të mëdha.

Një aspekt shumë i rëndësishëm gjatë programimit të këtij operacioni (përzgjedhjes) është kontrolli i trysnisë së përzgjedhjes. Nëse trysnia e përzgjedhjes është e madhe, atëherë bëhet përzgjedhja e individëve me gjasë shumë më të madhe dhe në këtë mënyrë bëhet shumimi i superindividëve. Në këtë rast algoritmi konvergjon para kohës drejt një optimizimi lokal. Në rast se trysnia e përzgjedhjes është e vogël, atëherë individët e mirë shumë ngadalë shumohen me ç'rast zgjidhjet e këqija nuk eliminohen nga popullata. Në këtë rast algoritmi konvergjon shumë ngadalë drejt zgjidhjes ose nuk konvergjon fare.

*Përzgjedhja Rank-based Roulette Wheel***:** Kjo skemë përzgjedhje funksion në atë mënyrë, ku fillimisht gjithë individët e popullatës radhiten në bazë të vlerës së funksionit për vlerësim (fitness) dhe mandej bëhet përllogaritje e gjasës së secilit nga këta individë. Për këtë arsye shkalla e presionit gjatë kërkimit është konstante dhe nuk ndikohet nga super-individët apo nga shpërndarja e vlerave të fitnessit siç ishte rasti te metoda Rank-based Roulette Wheel Selection.

Pjesë e këtij studimi (Noraini & Geraghty, 2011) është puna eksperimentale, ku autorët programuan dhe testuan algoritmin gjenetik në Matllab me instanca të ndryshme të problemit biznesmeni udhëtar, të marra nga TSPLIB, me 10, 20, 30 dhe 40 vende për t'u vizituar. Këtu u testua performanca e algoritmit gjenetik duke i përballur të tria metodat e përzgjedhjes dhe duke e testuar kështu algoritmin me këto instanca.

Gjetjet e këtij studimi eksperimental treguan se algoritmi gjenetik, i cili për metodë përzgjedhje shfrytëzon metodën Rank-based Roulette Wheel Selection jep zgjidhjet më cilësore, respektivisht turin më optimal (ose më të shkurtër) për të gjitha instancat dhe për testim, krahasuar me dy metodat e tjera të përzgjedhjes. E dyta radhitet Turnamet Selection, ndërsa e treta ose e fundit është Proportional Roulette Wheel Selection.

Për mendimin tim ky studim do të ishte më i vlefshëm sikur autorët të merrnin në konsideratë edhe parametra të tjerë për testim dhe që ndikojnë direkt në punën e algoritmit gjenetik, në rezultatin përfundimtar dhe në performancën e tij. Këta parametra janë: numri i gjeneratave, madhësia e popullatës dhe kohëzgjatja e ekzekutimit të algoritmit gjenetik. Kohëzgjatja është me rëndësi meqë është faktor që ka rëndësi si për përdoruesin, po ashtu edhe për vetë algoritmin. Sa më shumë kohë ekzekutimi të ketë në dispozicion algoritmi gjenetik, aq më shumë rritet mundësia që rezultati i tij të jetë më optimal (si rezultat të kemi një tur më të shkurtër). Në anën tjetër koha ishte me rëndësi për përdoruesin, meqë ai për shkak të kohës më të shkurtër zgjedh algoritmin gjenetik si alternativë kundrejt algoritmeve ekzakte për zgjidhjen e problemit biznesmeni udhëtar. Pra, sikur testet e kësaj pune eksperimentale, përveç metodave të përzgjedhjes, të kombinoheshin me njërin apo me gjithë parametrat e përmendur, të cilët ndikojnë direkt në punën e algoritmit, vlera ose cilësia e këtij studimi do të ngrihej dukshëm.

Rikombinimi

Përmes rikombinimit krijohen vetëm individ të rinj nga prindërit e përzgjedhur. Këtu duhet të bartet sa më shumë material gjenetik nga prindërit te pasardhësit. Tek algoritmi gjenetik rikombinimi konsiderohet si operacion parësor, përmes të cilit krijohen zgjidhje të reja. Gjasa e rikombinimit p_c përcakton se me ç'gjasë ndodh një rikombinim ose një individ i përzgjedhur, i pandryshuar, që bartet në popullatën e re. (Grubel, 2003)

Mutacion

Mutacioni tek algoritmi gjenetik konsiderohet si operacion dytësor dhe shërben që materialin e ri apo të humbur gjenetik ta risjellë përsëri në popullatë. Këtu ndryshohet gjeni i zgjedhur në mënyrë të rastësishme. Në përgjithësi ky operacion zbatohet shumë rrallë. Në të shumtën e rasteve gjasa e mutacionit përzgjidhet në atë mënyrë, kështu që në rast pritjeje, operacioni i mutacionit të ekzekutohet vetëm njëherë për kromozom. Pra, $p_m = \frac{1}{n}$ te një kromozom me gjatësi n (n-numri i gjeneve). (Grubel, 2003)

Vlerësimi

Duhet të jetë e mundshme që çdo kromozom të mund të vlerësohet me ndihmën e një funksioni, vlerësuar në varësi të problemit. Pra, ky funksion jep një dëshmi në lidhje me dobinë (përdorshmërinë) e zgjidhjes. Algoritmi gjenetik orvatet që individëve me dobi më të madhe t'u japë përparësi gjatë përzgjedhjes dhe në këtë mënyrë përmirëson popullatën.

Mbarimi (Termination)

Kriteri i mbarimit cakton pikën kohore se kur algoritmi duhet të përfundojë së ekzekutuari. Edhe këtu ka mundësi të ndryshme, prej të cilave vetëm disa vlejnë të përmenden. Pushimi i algoritmit mund të pasojë:

- Pas një numri të caktuar të gjeneratave.
- Sapo individi më i mirë nuk mund të përmirësohet pas një numri të caktuar gjeneratash.
- Sapo popullata apo një përqindje e saj të përbëhet nga individë identikë.
- Pas kalimit të një periudhe kohore.

Zgjedhja e kriterit të përfundimit varet nga problemi i dhënë dhe duhet të zgjidhet dhe të implementohet me kujdes. (Kämpf, 2006)

Thënë shkurt përmes algoritmit gjenetik programuesi orvatet që ecurinë e evolucionit që ndodh në natyrë ta simulojë në kompjuter, duke përdorur teknikat evolucionare të shumimit dhe mutacionit të individëve nga një popullatë fillestare e gjeneruar rastësisht dhe duke shpresuar kështu në krijimin e gjeneratave më të mira në lidhje me problemin e parashtruar.

Hybrid Mutation Genetic Algorithm

Në këtë punim (Katayama, Sakamoto, & Narihisa, 2000) autorët prezantojnë një algoritëm efecient gjenetik për zgjidhjen e problemit biznesmeni udhëtar, si problem kombinatorik për optimizim. Në përgjithësi, algoritmi gjenetik i përgjithshëm është konsideruar të jetë

qasje më e mirë sesa kërkimet lokale. Megjithatë, është e nevojshme për të forcuar aftësinë e kërkimit lokal, por dhe atë global, në mënyrë që të rritet efikasiteti i algoritmit gjenetik në përgjithësi.

Në këtë studim, algoritmi i tyre gjenetik zbaton një procedurë të ashtuquajtur "*stochastic hill climbing (SHC)*" në procesinmutacion (te procesi i mutacionit – ndryshimit të rastësishëm).

Gjetjet e autorëve gjatë këtij studimi treguan se një algoritëm i tillë gjenetik konvergjon deri në 99 % drejt rezultatit optimal, qoftë edhe për 500 vende për t'u vizituar.

Hybrid Mutation Genetic Algorithm (HMGA) përbëhet nga përzgjedhja e zakonshme dhe nga një operacion i ri crossover (bashkëdyzimi), përveç kërkimit lokal në operacionin mutacion (ndryshim i rastësishëm). Janë arritur zgjidhje praktike duke përdorur autorët, kërkimin lokal në vend të operacionit të zakonshëm të mutacionit. Për më tepër, Katayama dhe të tjerët aplikuan *Complete Subtour Exchange Crossover (CSE-X).*

Complete Subtour Exchange Crossover – Gjeneron thjesht disa trashëgimtarë përmes një procesi i cili shkëmben dhe kthen çdo subtour të zakonshëm në prindër dhe pastaj zgjedh dy ture nga ato që ka në dispozicion, bashkimi i të cilave formon gjatësi minimale.

Përshkrimi i algoritmit të propozuar nga Katayama, Sakamoto dhe Narihisa:

1) Kodimi për problemin TSP.
2) Gjenerimi i popullsisë fillestare.
3) Vlerësimi përmes funksionit për vlerësim.

4) Përzgjedhja.
5) Bashkëdyzimi (përmes CSE-X).
6) Mutacioni (përmes SHC).
7) Kontrolli i përmbushjes së kriterit për ndërprerje.

1) Kodimi i emrit të vendit për t'u vizituar dhe simboli si varg karakteresh sikur gjen apo kromozom. Ky varg karakteresh quhet individ, ndërsa gjatësia e tij është një numër i vendeve për t'u vizituar.
2) Gjenerimi i një numri të fundmë individësh në mënyrë të rastësishme, që quhet popullatë. Numri i individëve duhet të definohet para se algoritmi ta fillojë punën e tij.
3) Përmes funksionit për vlerësim llogaritet gjatësia e turit që është shuma e të gjithave distancave euklidiane në mes të vendeve për t'u vizituar.
4) Populllsia fillestare është zgjedhur në mënyrë arbitrare në mesin e individëve të mundshëm. Tek algoritmi i propozuar kjo përzgjedhje bëhet përmes strategjisë së shpjeguar në hapat prej 1 deri në 6 dhe siguron se anëtari më i mirë i popullatës mbijeton në gjeneratën e ardhshme.
5) Bashkëdyzimi përmes SHC-X, që dallon nga operacioni i njëjtë, tek algoritmi gjenetik i zakonshëm.
6) Zbatimi i metodës: "stochastic hill climbing", sepse është konstatuar të jetë algoritëm efecient për kërkim lokal.

7) Kontrolli se a është plotësuar kushti i paradefinuar për ndërprerjen e punës së algoritmit. Nëse kushti plotësohet, atëherë algoritmi pushon së ekzekutuari, përndryshe kërcen te hapi III.

Autorët tregojnë që algoritmin i lartëshënuar është programuar në gjuhën programuese C, dhe pas testimit kanë ardhur në përfundimin se: nëse zbatohet vetëm metoda e SHC-X, algoritmi do të konvergjojë shumë ngadalë, mirëpo nëse kësaj metode i shtohet edhe "stochastic hill climbing", siç vepruan autorët, konvergjimi drejt rezultatit është dukshëm më i shpejtë.

Algoritmi gjenetik dhe ueb shërbimet e Google-it

Autorët në punimin e tyre (Anupriya & Saxena, 2013) përdorën algoritmin gjenetik për të përcaktuar rrugën optimale në hartë Google-i (Google Map) dhe për zgjidhjen e problemit Biznesmeni udhëtar (Travelling Salesman Problem-TSP). Kjo zgjidhje është implementuar duke përdorur Google API dhe sistemin operativ Android. Të dhënat në lidhje me lokacionet (vendet) e ndryshme për t'u vizituar nxirren duke përdorur Google GeoCoder API. Pra, Anupriya & Saxena patën për qëllim të tregonin sesi të kombinohen algoritmi gjenetik dhe ueb shërbimet e Google-it për ta zgjidhur problemin TSP.

Anupriya dhe Saxena tregojnë përmes këtij punimi që përdorimi i ueb shërbimeve të Google-it është mjet efikas për zgjidhjen e problemit TSP, duke i ofruar kështu përdoruesit një ndërfaqe interaktive për zgjedhjen e vendeve për vizitë. Ata në punimin e tyre treguan, po ashtu, që duhet të ndërmerren disa hapa për të programuar një aplikacion Androidi i cili integron hartat e Google-it:

(i) Eclipse for Android Developers.

(ii) Instalimi i SDK-së.

(iii) Instalimi i Google Play Services.

(iv) Aplikimi përmes consoles së Google-it për marrjen e një çelësi, përmes të cilit lejohet qasje në ueb shërbimet e Google-it.

Ekzistojnë metoda numerike të ndryshme dhe që janë programuar algoritme të ndryshme për zgjidhjen e këtij problemit TSP. Bazuar në punimet krahasuese të të tjerëve, ku bëhet krahasimi i teknikave të programimit dinamik, deduktiv dhe atij gjenetik, Anupriya dhe Saxena vendosën për algoritmin gjenetik, meqë ky i fundit, sipas tyre, është provuar të jetë më i miri në zgjidhjen e problemit TSP me një numër të madh vendesh për t'u vizituar.

Sipas autorëve të punimit (Anupriya & Saxena, 2013), algoritmet gjenetike janë paradigma relativisht të reja në inteligjencën artificiale, që janë të bazuara në parimet e seleksionimit natyror. Teoria formale është zhvilluar fillimisht nga John Holland dhe nga studentët e tij në vitin 1970.

Ant Colony System (Algorimi kolonia e milingonave)

Milingonat arrijnë që ta eksplorojnë me sukses një territor të panjohur për të gjetur ushqim. Edhe pse ato janë praktikisht gati të verbëta dhe neve na duket që ato qarkullojnë pa kontroll. Çelësi i suksesit në komunikim qëndron te joshja apo tërheqja hormonale e ashtuquajtur feromon, të cilën në vazhdimësi çdo milingonë e tajitë dhe e liron. Nëse ndonjë milingonë gjatë kërkimit has në ndonjë burim ushqimi, ajo kthehet prapa për ta shenjuar rrugën që të çon deri tek ushqimi. Më pastaj ato orientohen në bazë të feromonit

që e kanë lëshuar vetë. Milingonat e tjera, që kullosin në atë sipërfaqe, do të hasin në feromonin e porsaliruar nga milingona e parë dhe atë do ta pasojnë. Kështu edhe milingonat e tjera e gjejnë burimin e ushqimit dhe kontribuojnë në mbledhjen dhe sjelljen e ushqimit deri te foleja e milingonave. Sa më shumë milingonat e bëjnë këtë, aq më shumë rrugë janë të shënuara deri tek ushqimi, derisa më në fund më shumë milingona ndjekin rrugën vajtje-ardhje – dhe jo çdo rrugë, por rrugën më të shkurtër, meqë feromoni nuk qëndron për një kohë të gjatë dhe vetëm ato rrugë mbetën për t'i pasur, të cilat janë frekuentuar më së shumti. Ky fenomen çon deri te gjetja e rrugës më të shkurtër e cila është paraqitur në figurën e mëposhtme. (Paul Theodor Pyl & Rudolph)

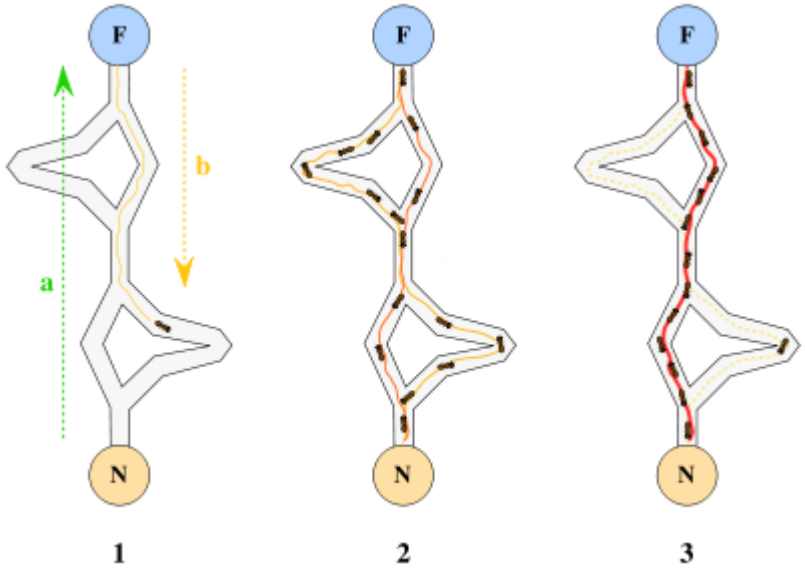


Figura 1: Analogjia e algoritmit të kolonisë së milingonave me atë si funksionojnë në natyrë

Ant Colony Optimization Algorithm (ACO)

Milingonat punojnë në grupe dhe eksplorojnë rrugën në kërkim për ushqimin e tyre dhe për të gjetur rrugën më të shkurtër për të arritur atje. Fillimisht, milingonat rastësisht lëvizin

përreth mjedisit që i rrethon në kërkim të ushqimit. Përderisa lëvizin përreth ato lirojnë një lloj kimikati në terren i cili është i njohur si feromon. Kur një milingonë e gjen ushqimin ajo kthehet prapa në folenë e saj, duke plotësuar në tokë përsëri sasinë e humbur të feromonit përgjatë atij shtegu. Për këtë shkak, milingonat e tjera e dinë se cili shteg apo rrugë duhet ndjekur për te burimi i ushqimit. Rruga e cila përmban sasi të lartë të shtigje feromone në terren është zgjedhur nga milingonat. Sa më e madhe është sasia e feromonit në një rrugë, aq më e madhe është mundësia që një milingonë ta përzgjedh atë rrugë. (Arora & Mamta, 2016)

Ant Colony Optimization Algorithm (ACO) është një teknikë probabilistike për zgjidhjen e problemeve kompjuterike e cila mund të reduktohet për të gjetur rrugë apo shtigje të mira përmes grafeve. Ky algoritëm është anëtar i familjes së algoritmeve të kolonisë së milingonave, po ashtu në metodat e tufave inteligjente dhe kjo përbën disa metoda meta-deduktive të optimizimit. (Darshni & Kaur, 2010)

ACO është teknikë e përgjithshme kërkimi bazuar në popullatë (bashkësi), për zgjidhjen e problemeve të vështira kombinatorike, e frymëzuar nga kolonitë reale të milingonave që gjurmojnë në bazë të feromonit të hedhur nga të tjerët. Sjellja e milingonave është shfrytëzuar në koloni të milingonave artificiale për kërkimin e zgjidhjeve të përafërta të problemeve diskrete të optimizimit dhe të problemeve të rëndësishme në telekomunikacion. (Darshni & Kaur, 2010)

ACO është një metodë meta-deduktive që bazohet në koloni artificiale milingonash që bashkëpunojnë (kooperojnë) mes veti për gjetjen e zgjidhjeve të mira, për probleme të vështira diskrete të optimizimit. Bashkëpunimi është një komponent kyç i familjes së algoritmeve të ACO-s. (Darshni & Kaur, 2010)

Zgjidhja është ndarja e burimeve të feromonit të këtyre agjentëve të thjeshtë artificialë të cilët komunikojnë në mënyrë të tërthortë me njëri-tjetrin. Zgjidhjet e kënaqshme janë veti e bashkëveprimit të ndërsjellët të këtyre agjentëve. Ideja origjinale prej kohësh është diverzifikuar për zgjidhjen e një game të gjerë problemesh numerike të vështira, dhe si rezultat dolën probleme të ndryshme, duke u bazuar në aspekte të ndryshme të sjelljes së milingonave. Ideja kryesore e inspiruar nga sjellja e milingonave është ajo e një kërkimi paralel mbi disa threads konstruktiv kompjuterik, bazuar në të dhënat e problemeve lokale dhe në një strukturë të kujtesës dinamike që përmban informacion mbi cilësinë e rezultatit paraprak. (Darshni & Kaur, 2010)

Autorët Dorigo & Gambardella në punimin e tyre (Dorigo & Gambardella, 1997) patën për qëllim paraqitjen e një pasqyre të përgjithshme të termave të huazuar nga natyra dhe përdorimin e tyre në programim. Pra, simulimin e punës dhe organizimin e kolonisë së milingona nga natyra në kompjuter. Në këtë punim një milingonë artificiale është një agjent i cili lëviz nga një qytet në tjetrin në një graf TSP-je. Ajo zgjedh një qytet, për të lëvizur përmes një funksioni probabilistik, i cili vlerëson gjurmët e grumbulluara të feromonit si dhe gjatësinë e rrugës. Milingonat artificiale preferojnë qytetet që janë të lidhura me rrugët me më shumë gjurmë feromone dhe që janë më afër. Fillimisht, milingonat artificiale janë vendosur në qytete të zgjedhura rastësisht. Në çdo interval kohor ato lëvizin drejt qyteteve të reja dhe modifikojë gjurmët e feromonit në rrugë e përdorura. Ky proces është quajtur përditësim lokal i gjurmëve. Kur secila nga milingonat të ketë përfunduar një tur të shkurtër, ajo ndryshon sasinë e feromenit përgjatë turit që asaj i përket. Mënyra sesi bëhet kjo, janë tri ide që mund t'i bartim te kolonia e milingonave artificiale duke analizuar sjelljen e natyrshme të milingonave natyrale:

(i) preferenca për shtigjet me një nivel të lartë feromoni,

(ii) shkalla më e lartë e rritjes së sasisë së feromonit në rrugë më të shkurtër, dhe

(iii) komunikimi mes milingonave i ndërmjetësuar përmes gjurmëve.

Dorigo & Gambardella gjithashtu shtuan se çelësi për zbatimin e algoritmit ACO për zgjidhjen e një problem të ri është të identifikojë një përfaqësim të përshtatshëm të problemit (si të paraqitet problemi si graf i cili mandej inspektohet nga shumë milingona artificiale), si dhe një metodë të përshtatshme për përcaktimin e distancës ndërmjet çdo dy nyjeve të grafit. Pastaj ndërveprimi i iniciuar në mesin e milingonave artificiale, përmes gjurmëve të feromonit të depozituar në brinjët e grafit (rrugën që lidh dy vende për t'u vizituar) gjeneron shpesh zgjidhje të kënaqshme, por jo optimale.

Algoritmi lakuriqët e natës

Autori Xin-She Yang në punimin e tij (Yang F. , 2010) prezanton një metodë të re për zgjidhjen e problemit TSP. Kjo metodë është metaheuristic e frymëzuar nga biologjia dhe bazohet në sistemin e orientimit të lakuriqëve. Në natyrë lakuriqët emetojnë ultra-tinguj pulsivë në mjedisin që i rrethon për shkak të orientimit dhe gjuajtjes së gjahut. Pas emetimit të këtyre pulseve, lakuriqët dëgjojnë jehonat, dhe në bazë të tyre ata lokalizojnë vendndodhjen e vet, identifikojnë pengesat dhe gjahun.

Ideja kryesore e këtij algoritmi të propozuar nga (Yang F. , 2010) është imitimi i këtij sistemi të lakuriqëve. Pra, lokalizimi përmes ultratingujve. Gjithsesi, disa rregulla të idealizuara duhet të merren parasysh në mënyrë që të bëhet një përshtatje e duhur:

- Të gjithë lakuriqët përdorin ultra-tinguj për të dalluar distancën dhe kanë “aftësi magjike’’ për ta dalluar prenë nga pengesat.
- Të gjithë lakuriqët fluturojnë rastësisht me një shpejtësi në një pozicion të caktuar, me një frekuencë të caktuar, gjatësi vale dhe volum të ndryshëm për të kërkuar një pre. Në këtë rregull të idealizuar, ne supozojmë se çdo lakuriq mund të rregullojë në mënyrë automatike frekuencën (ose gjatësinë valore) të ultra-tingullit të emetuar dhe shkallën e pulseve të emetuara. Ky korrigjim automatik varet nga afërsia e presë në shënjestër.
- Në situata reale, vëllimi i emetimeve te lakuriqët e natës mund të ndryshojë në shumë mënyra. Megjithatë, ne mendojmë se ky vëllim mund të ndryshojnë nga një vlerë e madhe pozitive në një vlerë minimale të vazhdueshme.

Të njëjtin algoritëm e prezantuan autorët në punimin e tyre (Osaba, Yang, Diaz, Lopez-Garcia, & Carballed, 2016), mirëpo kësaj radhe të përmirësuar. Në versionin klasik të algoritmit, të gjithë lakuriqët kryejnë lëvizjen e tyre duke ndjekur të njëjtin model gjatë gjithë fluturimit, pavarësisht nga pika në hapësirën në të cilën çdo lakuriq është i vendosur. Ndërsa në versionin e përmirësuar të këtij algoritmi, i jepet çdo lakuriqi të tufës një inteligjencë artificiale. Në këtë mënyrë, çdo lakuriq lëviz në një mënyrë të ndryshme në varësi të pozicionit të tij në lidhje me lakuriqët më të mirë të tufës.

Ky fakt në mënyrë të konsiderueshme rrit kapacitetin e eksplorimit të teknikës, duke çuar në një përmirësim të cilësisë së rezultateve.

Autorët Osaba dhe të tjerët, për të dëshmuar se propozimi i tyre është një metodë e përafërt dhe shumë premtuese për zgjidhjen e problemit TSP, krahasuan performancën e saj në 37

raste me rezultatet e arritura nga pesë teknika të ndryshme: algoritmi simulated annealing, algoritmi gjenetik, algoritmi gjenetik i shpërndarë, algoritmi xixëllonja dhe një algoritëm tjetër konkurrues. Po ashtu për krahasime të mirëfillta ata bënë edhe tri teste statistikore përgjatë punimit: Student's test, Holm's test dhe Friedman test.

Eksperimentimi i kryer në këtë studim ka treguar se algoritmi diskret i përmirësuar i lakuriqëve të natës, i tejkalon të gjitha alternativat e tjera në të shumtën e rasteve.

Ata, po në të njëjtin punim, thonë se: të dyja problemet TSP simetrike TSP josimetrike janë probleme standarde diskrete, dhe përkundër konkluzioneve të marra në këtë studim nuk mund të përgjithësohet për probleme të tjera të veçanta.

Sikur algoritmi i përmirësuar (modifikuar) i lakuriqëve të natës të krahasohej me metodat ekzakte dhe algoritmet komerciale për zgjidhjen e problemit biznesmeni udhëtar, do të bëhej hulumtim edhe më i dobishëm, sepse do t'ia sqaronte lexuesit qasjen e tyre në zgjidhjen e këtij problemi, edhe pse këto metoda nuk janë të ngjashme për kah filozofia dhe konceptet me algoritmin e propozuar nga Osaba dhe të tjerët.

2.9. Punime krahasuese bazuar në rezultate eksperimentale

Ilhan Ilhan bazuar në rezultatet eksperimentale të punimin të tij (Ilhan, 2016) erdhi në përfundimin se zgjidhja e problemit biznesmeni udhëtar përmes algoritmit kolonia e milingonave ACO mund të përdoret lehtë edhe te përcaktimi i rrugës te robotët autonomë (të pavarur). Po ashtu erdhi në përfundimin që ky algoritëm jep trajektore të njëjtë udhëtimi pa marrë parasysh sistemin operativ dhe mjedisin ku testohet; për vende të njëjta për t'u vizituar..

Pra, zgjidhja e propozuar (Ilhan, 2016) është një sistem për planifikimin e rrugës bazuar në një microcontroller i cili posedon një procesor: ARM Cortex - M4. Sistemi (hardueri) i propozuar importon informatat në kohë reale në lidhje me lokacionet përmes një moduli të GPS-it i cili përmes një porti serik komunikon me microcontrollerin. Algoritmi i kolonisë së milingonave është programuar për zgjidhjen e problemit biznesmeni udhëtar dhe kodi burimor i tij është ruajtur në këtë microcontroller. Kodi burimor i algoritmit u programua nga autori përmes mikroC PRO, mjet ky për programimin e ARM-së. Microcontrolleri kishte këto karakteristika: shpejtësia prej 168 MHz, 1 Mbyte flash memorie të programueshme dhe 192 Kbytes RAM.

Në të njëjtën kohë ky algoritëm u programua nga Ilhan Ilhan edhe për kompjuter. Për t'i futur informatat e duhura (gjatësinë dhe gjerësinë gjeografike) të çdo vendi në kompjuter, u programua një ndërfaqe për përdorues përmes Microsoft Visual C#. Kompjuteri ku u testua programi kishte një procesor I7 prej 2.4 GHz –sh, 8 GB RAM dhe sistem operativ Windows Home 8.1.

Algoritmi ACO u testua si në microcontoller, ashtu edhe në kompjuter me të dhëna reale, fillimisht me 10 vende për vizitë në Turqi, dhe më pas 15 vende për vizitë nga rrethina e provincës Konja. Ky algoritëm u testua po ashtu për nga përformaca, që nënkupton kohën e ekzekutimit dhe optimizimin e rrugës në kilometra. Pasi rezultatet e përfituara u paraqitën në formë tabelore, autori erdhi në përfundimin se pavarësisht dallimeve në kohën e ekzekutimit, algoritmi dha në të dyja mjediset për testim të njëjtën rrugë optimale.

Meqë algoritmi funksionoi pa problem në microcontroller, po me të njëjtin microcontroller mund të bartet edhe përcaktimi i rrugës për sisteme të tjera si: robot të pavarur dhe mjete fluturimi.

Do të kishte qenë hulumtim i mrekullueshëm sikur të programohej algoritmi kolonia e milingonave, që të funksiononte edhe në pajisje të tjera mobile, siç janë smart – phone-ët, dhe testohej se a jep të njëjtin rezultat, respektivisht të njëjtin tur për të njëjtat vende për t'u vizituar.

Autorët Nodehi, Fadaei dhe Ebrahimi në punimin e tyre (Nodehi, Fadaei, & Ebrahimi, 2016) propozuan një algoritëm të ri, të quajtur **r**andomized **g**ravitational **e**mulation **s**earch algorithm (RGES) për zgjidhjen e problemit TSP. Ky algoritëm është i bazuar në konceptet e kërkimit të rastësishëm duke përdorur dy nga katër parametrat kryesorë të shpejtësisë dhe forcës së gravitetit në fizikë.

Rezultatet e punës eksperimentale treguan se algoritmi RGES - TSP kërkon shumë më pak kohë për të arritur rezultate më të mira në krahasim me algoritmin gjenetik GA. Rezultatet e këtij algoritmi të përmirësuar bëhen më të dukshme me rritjen e numrit të vendeve për t'u vizituar.

Pra, Nodehi, Fadaei dhe Ebrahimi erdhën në këto përfundime pasi testuan algoritmet e programuara në një kompjuter me këto karakteristika: Procesori 2.4 GHz i tipit pentium IV dhe 1 GB RAM. Programimi i këtyre dy algoritmeve është bërë në Matlab. Autorët prezantuan në mënyrë tabelore performancën e algoritmeve duke i testuar ata me të dhëna të ndryshme për testim: me 20 vende për testim, 50, 100, 500 dhe 1000 vende për testim. Koha e ekzekutimit dhe gjatësia e rrugës së optimizuar janë po ashtu parametrat që u testuan. Pra, në mënyrë tabelore u paraqitën rezultatet e përfituara në rastin më të mirë dhe në rastin më të keq i të dyja algoritmeve.

Meqë përparësia e algoritmit të quajtur **r**andomized **g**ravitational **e**mulation **s**earch bëhet gjithnjë e më e dukshme me rritjen e numrit të vendeve për t'u vizituar si dhe që këtij algoritmi i duhet kohë më e shkurtër ekzekutimi, në krahasim me algoritmin gjenetik, atëherë do të ishte e udhës që këto dy algoritme të kombinohen dhe të formohet një algoritëm hibrid për zgjidhjen e problemit biznesmeni udhëtar.

Në punim (Arora & Mamta, 2016) u prezantua grafikisht puna e algoritmit gjenetik dhe atij kolonia e milingonave për optimizim (**A**nt **C**olony **O**ptimization - ACO) përmes grafeve në rrafshin XY. Autorët përdorën për programim një laptop, ku ishte i instaluar Windows 7. Laptopi kishte një procesor I5 me 2.5 GHz dhe 4 GB RAM. Programimi i të dy algoritmeve është bërë në Matlab.

Grafi me ngjyrë të kuqe demonstronte punën e algoritmit gjenetik, ndërsa ai me ngjyrë të kaltër punën e algoritmit ACO. Çdo vend për t'u vizituar paraqitet si pikë në rrafsh dhe adresohet përmes koordinatave (X, Y), ndërsa brinjët e grafit janë rrugët që lidhin ato vende. Thënë shkurt autorët testuan perfomancën e algoritmeve AG dhe ACO në dy aspekte: cilësinë e rezultatit (cili nga ata optimizon rrugën më tepër) dhe kohëzgjatjen e ekzekutimit. Algoritmet u testuan me tri instanca të ndryshme: me 14 vende për t'u vizituar, 30 vende për t'u vizituar dhe 50 vende për t'u vizituar. Nga rezultatet e përfituar autorët ndërtuan dy tabela që ta kenë më të lehtë krahasimin e algoritmeve, ku njëra përmbante gjatësinë e turit pas optimizimit të secilit algoritëm dhe për secilën instancë për testim, ndërsa tjetra përmbante kohët që iu deshën secilit algoritëm për optimizimin e turit për instancë të caktuar.

Në punim (Arora & Mamta, 2016) u prezantua grafikisht puna e algoritmit gjenetik dhe atij kolonia e milingonave për optimizim (**A**nt **C**olony **O**ptimization - ACO) përmes grafeve në rrafshin XY. Autorët përdorën për programim një laptop, ku ishte i instaluar Windows 7. Laptopi kishte një procesor I5 me 2.5 GHz dhe 4 GB RAM. Programimi i të dy algoritmeve është bërë në Matlab.

Grafi me ngjyrë të kuqe demonstronte punën e algoritmit gjenetik, ndërsa ai me ngjyrë të kaltër punën e algoritmit ACO. Çdo vend për vizitë paraqitet si pikë në rrafsh dhe adresohet përmes koordinatave (X, Y), ndërsa brinjët e grafit janë rrugët që i lidhin ato vende. Thënë shkurt autorët testuan perfomancën e algoritmeve AG dhe ACO në dy aspekte: cilësinë e rezultatit (cili nga ata optimizon rrugën më tepër) dhe kohëzgjatjen e ekzekutimit. Algoritmet u testuan me tri instanca të ndryshme: me 14 vende për vizitë, 30 vende për vizitë dhe 50 vende për vizitë. Nga rezultatet e përfituar autorët ndërtuan dy tabela që ta kenë më të lehtë krahasimin e algoritmeve, ku njëra përmbante gjatësinë e turit pas optimizimit të secilit algoritëm dhe për secilën instancë për testim, dhe tjetra përmbante kohët që ju deshën secilit algoritëm për optimizimin e turit për instancë të caktuar.

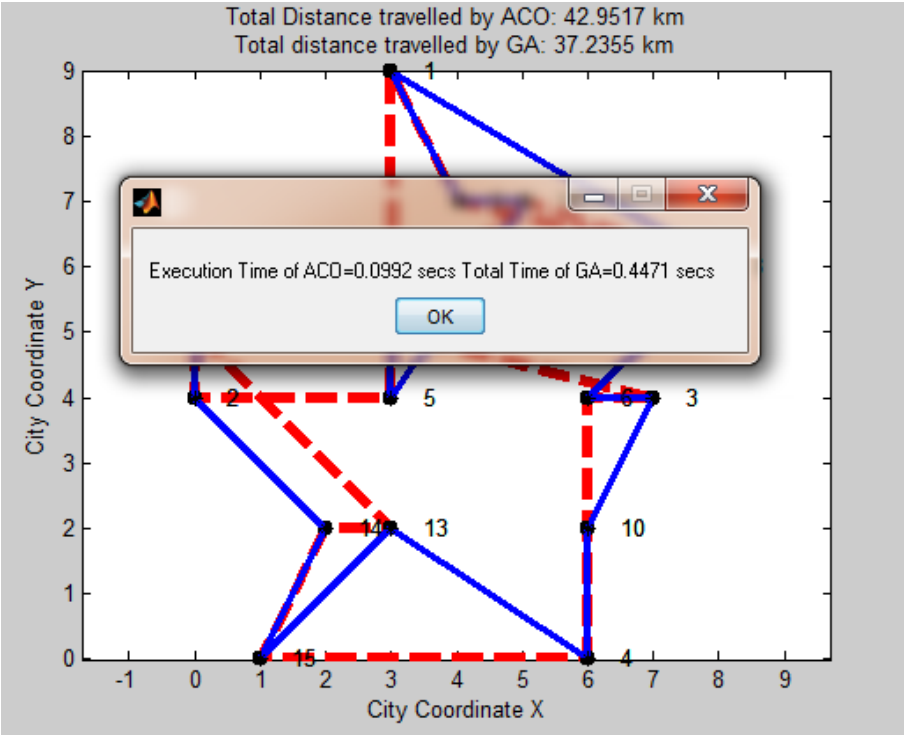


Figura 2: GUI i algoritmit gjenetik dhe kolonisë së milingonave

Gjetjet apo rezultatet e përfituara treguan që secili nga këto algoritme deduktive është i aftë që ta zgjidhë problemin e biznesmenit udhëtar (Travelling Salesman Problem). Krahasimi midis dy është i bazuar në numrin e qyteteve të dhëna në këtë problem dhe pastaj përcakton rezultatin në bazë të distancës së turit nga ana e tyre dhe kohën e ekzekutimit. Autorët erdhën në përfundim se algoritmi gjenetik optimizon rrugën më shumë sesa algoritmi ACO nëse është fjala për një numër më të vogël të vendeve për t'u vizituar, ndërsa algoritmi ACO kryeson nëse numri i vendeve për t'u vizituar është më i madh. Ndërsa sa i përket kohës së ekzekutimit, algoritmi ACO ka gjithnjë kohë më të shkurtër se algoritmi gjenetik, pa marrë parasysh numrin e vendeve për t'u vizituar. Pra, ACO ka performancë më të mirë kohore krahasuar me algoritmin gjenetik.

Këtu u krahasuan që të dyja algoritmet në bazë të distancës totale të turit si dhe të kohës së ekzekutimit për një numër të caktuar vendesh për t'u vizituar, kishte qenë më mirë që këto algoritme të testoheshin po ashtu edhe në bazë të kostos totale. Pra, në këtë mënyre do të mund të klasifikonim se cili nga këto algoritme është më efiçent, bazuar në koston totale, meqë në shumicën e punimeve algoritmet janë testuar në të njëjtën mënyrë. Pra, sikur të testohej efiçenca e algoritmeve, bazuar në koston totale, do ta bënte këtë punim që në thelb të dallohej nga të tjerët.

Në këtë punim (Adewole, Otubamowo, & Egunjobi) autorët tregojnë se algoritmet metaheuristic-e kanë provuar të jenë zgjidhëse të mira për problemin biznesmeni udhëtar (TSP). Në shkencat kompjuterike me termin 'metaheuristic' iu referohen metodave llogaritëse për zgjidhjen e problemeve që kanë të bëjnë me optimizimit, të cilat orvaten që hap pas hapi të përmirësojnë zgjidhjet e mundshme të problemit për kah aspektit cilësor deri te një zgjidhje e kënaqshme.

Autorët Adewole, Otubamowo & Egunjobi implementuan dy algoritme: gjenetik dhe atë Simulated Annealing për zgjidhjen e problemit TSP.

Punimi i tyre përqendrohet te krahasimi i këtyre dy algoritmeve bazuar në rezultatet dhe performancën e tyre gjatë zgjidhjes së problemit TSP. Pra, fokus të veçantë ka mënyra e krahasimit të këtyre algoritmeve si dhe parametrat nga të cilët varet puna e tyre si dhe plotësimi i kriterit për të pushuar së ekzekutuari.

Krahasimit i tyre është bërë bazuar në dataset, cilësinë e zgjidhjes, në parametrat dhe kohën e ekzekutimit.

a) Dataset- grupe të dhënash për testim që janë marrë nga disa qytete kryesore nigeriane, disa janë marrë nga www.bosn.org dhe disa të tjera janë gjeneruar në mënyrë të rastësishme.

b) Parametrat hyrës që u përdorën për secilin algoritëm janë:

Algoritmi gjenetik	**Algoritmi simulated annealing**
Popullata	Temperatura
Shkalla e ndryshimit	Shkalla e ftohjes
Gjatësia e prerjes	Temperatura absolute

Tabela 3: Parametrat e përdorur nga të dy algoritmet (gjenetik dhe simulated annealing)

c) Koha e ekzekutimit

Faktori kohë ka rëndësi jetike te metodat metaheuristic-e. Koha është arsyeja kryesore; është ajo që ne i japim përparësi algoritmeve që japin zgjidhje joekzakte por të kënaqshme, ndaj atyre që japin rezultat ekzakt.

d) Krahasimi i cilësisë së zgjidhjes është më se i domosdoshëm, meqë është e rëndësishme të dihet se cili nga algoritmet jep rezultat më të përafërt me atë ekzaktin. Pra, cili nga ata na jep rezultat më të kënaqshëm?

Pra, bazuar në punën krahasuese dhe në rezultatet eksperimentale, autorët Adewole, Otubamowo dhe Egunjobi erdhën në përfundimin që algoritmi simulated annealing ka kohë më të shkurtër ekzekutimi sesa algoritmi gjenetik. Algoritmi gjenetik edhe pse jep rezultate më të kënaqshme sesa simulated annealing, nuk është i përshtatshëm nëse madhësia e popullatës është e madhe, si parakusht për zgjidhje më cilësore. Sido që të jetë, që të dy

algoritmet janë algoritme të përshtatshme për zgjidhjen e problemit TSP nëse njeriu di t'i vendosë parametrat e përshtatshëm para se algoritmet të fillojnë punën e tyre.

Për përpunimin e një numri të vogël vendesh për t'u vizituar autori thotë se TSP-Brute Force-Algoritmi është i mjaftueshëm si algoritëm joefikas, mirëpo për problemet ku përfshihet një numër i madh i vendeve për t'u vizituar duhet të përdoren algoritme më të sofistikuara, të cilat japin rezultate të kënaqshme brenda një kohe të caktuar. (Stanec, 2011)

Roman Stanec në hulumtimin e tij (Stanec, 2011) kishte për qëllim gjetjen e algoritmeve efikase të cilat mund të përdoren në zgjidhjen e problemit TSP në kushte të kufizuara: kohë të kufizuar dhe fuqi llogaritëse kompjuterike të kufizuar (jo te ndonjë super-kompjuter). Pasi krahasoi rezultatet e algoritmeve të përzgjedhura, atëherë implementoi një zgjidhje e cila është në gjendje të kthejë si rezultat turin më të mirë të mundshëm. Algoritmet e përzgjedhura nga autori janë tri: branch-and-bound, greedy dhe nearest neighbour.

Algoritmi Branch-And-Bound bën pjesë në algoritmet ekzakte për zgjidhjen e problemit TSP, dhe njëkohësisht është mjeti më i përdorur për zgjidhjen e problemeve kombinatorike që nuk zgjidhen në kohë polinomiale (NP-hard). Algoritmet Greedy dhe Nearest neighbour i takojnë algoritmeve deduktive (heuristic) dhe sugjerojnë në përgjithësi zgjidhje të përafërta në lidhje me problemet e optimizimit. Thënë shkurt këto algoritme janë të afta të japin rezultate te pranueshme në lidhje me zgjidhjen e problemin TSP, mirëpo për to nuk ka dëshmi formale në lidhje me korrektësinë e tyre. Çdo zgjidhje deduktive e TSP-së mund të vlerësohet në aspektin e dy parametrave kryesorë: kohën e ekzekutimit dhe cilësinë e turneve që ajo prodhon. Algoritmi Greedy mund të programohet në atë mënyrë që koha e

ekzekutimit të tij të jetë: $\theta(N^2 \log N)$, ndërsa e algoritmit Nearest neighbour është: $\theta(N^2)$. (Stanec, 2011)

Pas rezultateve të përfituara eksperimentuese dhe diskutimeve të bëra në lidhje me to, Roman Stanec programoi një sistem për zgjidhjen e këtij problemi në gjuhën programuese C, bazuar në një arkitekturë specifike, mirëpo pa Graphical User Interface (GUI). Kjo arkitekturë përbëhet nga tri komponentë kryesorë:

1) Dicision Maker – vendos në bazë të dhënave që jep përdoruesi, se cili nga tri algoritmet e lartpërmendura është më i përshtatshëm dhe për zgjidhje të problemit, ekzekuton atë. Të dhënat jepen nga përdoruesi përmes Command-line.
2) Bërthama e sistemit ku rrinë tri algoritmet për ekzekutim.
3) Gjeneratori i OutPut-it – i cili merr rezultatin nga algoritmi i caktuar dhe ia shfaq atë përdoruesit në mënyrë të formatuar në Command-line. Pra, numri i vendeve për t'u vizituar, vargu i koordinatave që duhet ndjekur, si dhe gjatësia e rrugës në kilometra.

Në bazë të rezultateve të testimit, Stanec ka ardhur në përfundim që algoritmi Branch-And-Bound jep rrugën më optimale, mirëpo është i ngadalshëm, dhe për numër më të madh të vendeve për t'u vizituar është përafërsisht i papërdorshëm. Për numër të madh të vendeve për t'u vizituar duhet të përdoren dy algoritmet e tjera, ku Greedy-Algoritmi jep mesatarisht 14.4 % rezultat më të keq sesa ai më optimali, ndërsa algoritmi Nearest neighbour jep mesatarisht 16% rezultat më të keq sesa ai më optimali. Fakti kryesor për Dicision Maker është ai që shpejtësia e llogaritjes është më e madhe nëse ai vendos të përdorë Nearest

neighbour, ndërsa nëse përdor Greedy-algoritmin jep rezultat më optimal ku gjatësia e rrugës është më e shkurtër.

Krahasim i precizitetit në mes OpenStreetMap, Bing Maps dhe Google Maps në lidhje me Irlandën

Në hulumtimin e tyre (Cipeluch, Jacob, Winstanley, & Mooney, 2010) autorët: Cipiluch dhe të tjerët prezantuan një krahasim saktësie të hartave: OpenStreetMap, Bing Maps dhe Google Maps në rastin e Irlandës. Autorët përzgjodhën 5 qytete dhe vende si raste studimi. Ata analizuan secilin nga këto tri sisteme hartash për mbulim hapësinor, saktësi dhe saktësi pozicioni në terren. Gjetjet e këtij punimi konstatuan, se edhe pse asnjëra nga këto platforma hartash nuk ishte fituese e qartë, ato tregojnë dallime dhe ngjashmëri mes veti. Ata janë të mendimit që rezultatet e përfituara nga hulumtimi i tyre ndihmojnë një kategori të caktuar personash, të cilët zhvillojnë shërbime të bazuara në vendndodhje për vendet si Irlanda dhe të tjera, ku qasja në të dhënat gjeo-hapësinore me cilësi të lartë shpesh është shumë e shtrenjtë ose bëhet e vështirë nga barriera të ndryshme siç janë: mungesa e të dhënave ose kufizimeve të qasjes.

Nëse dëshirojmë të zgjedhim qasje falas, në ndonjë sistem hartash për navigim, bazuar në ueb, apo informata gjenerale, ekzistojnë tri burime më të popullarizuarat të tilla si: OpenStreetMaps, Google Maps dhe Bing Maps. Pra, këto tri platforma hartash bazuar në ueb janë më të popullarizuarat për shfrytëzim te shërbimet e bazuara në vendndodhje me theks të veçantë për navigimin e këmbësorëve, për aplikimet që udhëheqin turistët si dhe për aplikime të tjera kërkimi bazuar në vendndodhje.

Autorët i qasen problemit në këtë mënyrë:

Të dhënat e Open Street Map për Irlandën ata i shkarkuan në formatin OSM-XML në marsin e vitit 2010 nga Cloudmade. Me ndihmën e veglës OSM2PGSQL i importuan ato në bazë të dhënash PostGIS. Kjo i dha qasje autorëve në të gjitha të dhënat e OSM-së për Irlandën në një bazë të dhënash relacionale. Ndërsa mjetin org2org nga libraria GDAL ata e përdorën që t'u ofrojë një funksionalitet, që përmes comand-line rezultatet e SQL-Queries të konvertohen në format gjeo-hapësinor si fajlla: Shape dhe KML. Ata dizajnuan një grup SQL-Queries që të performojnë sendet e mëposhtme:

(i) Ekstraktimi i gjithë rrugëve, korsive etj. në formatin KML për secilin rast studimi

(ii) Ekstraktimi i të gjitha POI-ve në formatin KML.

POI-të ishin më tej të ndara dhe të klasifikuara në hotele, qendra tregtare etj. Ata ngarkuan me lehtësi POI – fajllat në Google Maps dhe Bing Maps me ndihmën e një ueb aplikimi që zhvilluan duke përdorur OpenLayer që u lejoi atyre parjen e KML – fajllave të ngarkuar.

Në këtë punim Cipiluch dhe të tjerët testuan këto tri platforma hartash në lidhje me këto vende: Ennis, Drogheda, Maynooth, Waterford dhe Dublin. Autorët në atë kohë numëruan rrugët: autostradat, rrugët kryesore nacionale, rrethrrotullimet në rrugët kryesore nacionale, rrugët regjionale, rrethrrotullimet në rrugët regjionale, rrugët në pronat e banimit dhe rrethrrotullimet në pronat e banimit.

Në formë tabelore përmblodhën dhe më pas interpretuan rezultatet. Atëbotë erdhën në përfundim se në secilën nga këto 5 raste studimi nuk ka saktësi të qëndrueshme të këtyre tri platformave të hartave.

Gjetjet treguan që kishte emra jokorrektë rrugësh dhe zgjerime rrugësh te Bing Maps dhe Google Maps, të cilat dështonin brenda qendrës së qytetit. P. sh.: N11 te rasti për testim i Dublinit.

1) Te Google Maps ishte i vështirë filtrimi i rezultateve kur kërkohej për hotele.
2) Bing Maps në rastin e studimit Dublin ka shumë më pak mbulesë territore sesa të tjerat.
3) Atëkohë hulumtimi autorët erdhën në përfundim gjithashtu se OpenStreetMaps është i lidhur ngusht me numrin e vullnetarëve që punojnë në vende dhe zona të ndryshme.

Ne mendojmë se, që OSM të merret si një burim serioz i të dhënave metrike hapësinore, duhet të zhvillohet për të lejuar matjen e të dyjave: saktësisë dhe mbulimit në nivel lokal,vendor dhe në nivelet e qarkut. Lejimi i matjes së këtyre të dhënave duhet të jetë sasior, në mënyrë që ato të pranohen (dhe të përdoren) gjerësisht nga gjeo-komuniteti.

2.10. Përmbledhje

Ky kapitull është kapitulli bazë i këtij punimi. Këtu është shpjeguar origjina e problemit biznesmeni udhëtar, është bërë definimi i problemit dhe klasifikimi i tij. Në këtë kapitull është bërë po ashtu formulimi matematik i problemit, meqë është gjë e pashmangshme nëse dëshirojmë ta zgjidhim këtë problem. Formulimi matematik i këtij problemi është bërë përmes teorisë së grafeve, meqë janë mjeti më i përshtatshëm për formulimin e këtij problemi dhe se ekzistojnë algoritme që zbatohen mbi to. Po ashtu, është bërë një hyrje në teorinë e grafeve, ku sqarohen konceptet themelore në lidhje me to, për ta bërë sa më të lehtë kuptimin e formulimit matematik të këtij problemi. Ky problem është i formuluar edhe si programim linear. Për ta treguar rëndësinë e këtij problemi janë cekur dhe disa aplikime ku gjen zbatim ky problem, si dhe problemet e ngjashme me të: kabllimi në kompjuter (Computer wiring), rrugëtimi i automjeteve-kamionëve (Vehicle Routing),

rrugëtimi i autobusëve shkollorë, procesi i shpimit të pllakave elektronike, Design of global navigation satellite system surveying networks (GNSS).

Metodat numerike (algoritmet) për zgjidhjen e problemit biznesmeni udhëtar ndahen në ekzakte, pra në ato që japin zgjidhje ekzakte dhe në ato joekzakte apo që japin zgjidhje joekzakte.

Në algoritmet ekzakte bëjnë pjesë: Branch and Bound dhe Branch and Cut si dhe programimi dinamik. Këtu sqarohen idetë themelore, principet e punës së tyre si dhe diskutohet sipërfaqësisht rreth kohës së ekzekutimit të tyre.

Algoritmet joekzakte ndahen në të përafërta dhe deduktive. Këtu bëjnë pjesë: Simulatead Annealing, Algoritmi Gjenetik dhe Ant Colony System. Algoritmit Gjeneti dhe atij Simulated Annealing i kushtohet më tepër rëndësi meqë janë algoritme që përdoren për zgjidhjen e problemit dhe tjetri si algoritëm alternativ dhe për krahasim.

Meqë produkti softuerik është kompleks dhe kombinon në vete sisteme të ndryshme, atëherë në kapitullin vijues do të bëhet fjalë për një teknologji të ashtuquajtur ueb shërbimet. Ueb shërbimet përcaktojnë dhe sigurojnë që infrastruktura IT të lejojë aplikime të ndryshme që të shkëmbejnë të dhëna dhe të marrin pjesë në proceset e biznesit pavarësisht nga sistemet operative ose nga gjuhët e programimit të këtyre aplikimeve.

KAPITULLI III: ARKITEKTURA E ORIENTUAR DREJT SHËRBIMEVE (SOA)

3.1. Hyrje

Arkitekturat dominuese të aplikacioneve klient/server dhe shumështresore të dekadave të fundit kanë ngritur një numër të madh problemesh të cilat lidhen me përshkallëzimin dhe disponueshmërinë e sistemeve IT. Të tjera probleme u shfaqën gjatë fazës për të inkorporuar sistemet e vjetra në ato të reja. (Fejzaj, 2014)

Arkitektura e orientuar drejt shërbimeve (**S**ervice **O**riented **A**rchitecture SOA) është një stil arkitekturor që udhëheq të gjitha aspektet e krijimit dhe të përdorimit të proceseve të biznesit të paketuara si shërbime. Gjithashtu SOA përcakton dhe siguron që infrastruktura IT të lejojë aplikime të ndryshme që të shkëmbejnë të dhëna dhe të marrin pjesë në proceset e biznesit pavarësisht nga sistemet operative ose gjuhët e programimit të këtyre aplikimeve. SOA përfaqëson një model në të cilin funksionaliteti dekompozohet në njësi të vogla të veçanta (shërbime), të cilët mund të shpërndahen në rrjet dhe mund të kombinohen së bashku, por dhe të ripërdoren për të krijuar aplikime biznesi. Këto shërbime komunikojnë me njëri-tjetrin duke kaluar të dhënat nga një shërbim tek tjetri, ose duke koordinuar një aktivitet ndërmjet një ose më shumë shërbimesh. (Fejzaj, 2014)

Google është gjiganti i cili ofron ueb shërbime interaktive në formë hartash për përdorues të ndryshëm. Këto ueb shërbime ofrohen nga Google bazuar në standardin RESTful. Ky është njëri nga dy standardet, që ofrohen ueb shërbimet e ditëve të sodit.

Meqë të gjitha ueb shërbimet e Google-it ofrohen në Cloud dhe ato përdorën për arritjen e objektivave të kësaj doktorate, e shoh të domosdoshme futjen e një nënkapitulli për ta sqaruar këtë teknologji moderne. Po ashtu, puna praktike e kësaj doktorate ka të bëjë me programimin e një aplikimi për pajisje mobile, që konsumon ueb shërbimet e Google-it, del të jetë i paevitushëm sqarimi i teknologjisë Mobile Cloud Computing.Teknologji e cila përdor teknikat e cloud computing për ruajtjen dhe përpunimin e të dhënave për pajisjet mobile.

3.2. Komponentët arkitekturore të SOA-së

Komponentët bazë të një arkitekture të orientuar nga shërbimet janë:

1. Ofruesi i shërbimeve (Sevice providers): Një ofrues shërbimesh është një komponent ose një bashkësi komponentësh që ekzekuton funksionet e biznesit, gjithashtu pranon inpute dhe outpute. (Fejzaj, 2014)

2. Konsumuesi i shërbimeve (Service consumer): Një konsumues shërbimesh është një bashkësi komponentësh të interesuar të përdorin shërbimet që mundësohen prej ofruesve të shërbimeve. (Fejzaj, 2014)

3. Depozituesi i shërbimeve (Service repository): Një depozitues përmban përshkrimet e nevojshme për të gjitha shërbimet që ofrohen nga shërbyesit e ndryshëm. Ofruesit e shërbimeve i depozitojnë shërbimet që ata ofrojnë në këta komponentë dhe konsumuesit e tyre mund t'i përdorin ato kur ata kanë nevojë. (Fejzaj, 2014)

3.3. Ueb shërbimet

Ueb shërbimet (Web Services) janë një teknologji e re që mundëson qasjen e funksioneve në distancë përmes Internetit. Ueb shërbimet janë aplikacione kyçe në komunikimin e bizneseve me biznese (B2B) dhe biznesit me konsumator (B2C). Ueb shërbimet janë të bazuara në një bashkësi XML-standardesh, të cilat janë WSDL (Web Services Descritpion Language), SOAP (Simple Object Access Protocol) si dhe UDDI (Universal Descripption, Discovery and Integration). Përmes këtyre standardeve ueb shërbimet mundësojnë që nga një Windows klient t'i qasemi një funksioni në distancë, i cili ofrohet nga një sistem në Linux dhe anasjelltas. Komunikimi në mes të shfrytëzuesit të ueb shërbimit (klientit) dhe ofruesit te ueb shërbimeve bëhet përmes HTTP (Hyper Text Transfer Protocol) protokollit. Komunikimi përmes HTTP-së (portit 80) protokollit nuk implikon ndërrimin e konfiguracionit të rrjetës, meqenëse zakonisht porti 80 është në çdo organizatë i hapur, çka e ka rritur shkallën e pranueshmërisë së kësaj teknologjie te konsumatori. (ZENUNI, 2007)

Bota e teknologjisë po bëhet gjithnjë e më konkurruese me zhvillimin e shpejtë të metodave të reja për zgjidhjen e problemeve të përbashkëta, që i hasin institucionet e ndryshme. Kompanitë gjigante të zhvillimit të softuerëve dhe hulumtuesit e ndryshëm promovojnë idetë e tyre për të lehtësuar punën e organizatave të biznesit dhe të atyre jofitimprurëse; ata tentojnë të bëjnë një mjedis të teknologjisë së informacionit ku një plejadë e kompjuterëve personalë, pajisjeve të mençura, serverëve dhe tabletave që të mund të komunikojnë me njëri-tjetrin pa pengesa dhe kjo është e mundur me zbulimin e ueb shërbimeve. (Halili, 2014)

Kompanitë e biznesit, institucionet qeveritare dhe organizatat jofitimprurëse duke përdorur ueb shërbimet janë në gjendje të ndajnë të dhënat, të integrojnë proceset e tyre, si dhe të bashkojnë forcat për të siguruar zgjidhje të plotë dhe të përshtatshme për klientët e tyre. (Halili, 2014)

Shtresat e teknologjisë së ueb shërbimeve

Arkitektura e ueb shërbimeve është e ndërtuar në pesë shtresa kryesore, të cilat janë të vendosura njëra pas tjetrës ose njëra mbi tjetrën:

Figura 3: Shtresat e teknologjisë së ueb-shërbimeve *(Tidwell, Snell, & Kulchenko, 2001)*

Discovery

Kjo shtresë ofron një mekanizëm përmes të cilit informohet shfrytëzuesi i ueb-shërbimit me përshkrimet që i ka bërë ofruesi ueb-shërbimit të tij. Një ndër mekanizmat më të njohur që kryen këtë funksion është **U**niversal **D**escription, **D**iscovery and **I**ntegration (UDDI). (Tidwell, Snell, & Kulchenko, 2001)

Description

Kjo është shtresa ku bëhet përshkrimi i ueb shërbimit kur ai të implementohet (programohet). Një ndër gjuhët apo standardet më të njohura përmes të cilit bëhet përshkrimi i ueb shërbimit është Web Services Description Language (WSDL). (Tidwell, Snell, & Kulchenko, 2001)

Packaging

Të dhënat e aplikacioneve për t'u transmetuar nëpër rrjetë nga shtresa e transportit, ato duhet të paketohen në atë mënyrë që të kuptohen nga të gjitha palët pjesëmarrëse. (Tidwell, Snell, & Kulchenko, 2001)

Transport

Shtresa e transportit përfshin teknologjitë e ndryshme që mundësojnë komunikimin e drejtpërdrejtë të aplikacioneve mes veti. Teknologji të tilla përfshijnë protokollet si TCP, HTTP, SMTP dhe Jabber. Roli kryesor i kësaj shtrese është transmetimi i të dhënave në mes dy lokacioneve në rrjetë. (Tidwell, Snell, & Kulchenko, 2001)

Network

Roli i kësaj shtrese është i njëjti sikur i shtresës së rrjetës te modeli TCP/IP, pra kjo siguron aftësitë bazë për komunikim, siç janë: adresimi dhe rrugëtimi.

Arkitektura e ueb shërbimeve

Ueb shërbimet sigurojnë mundësi komunikimi ndërmjet sistemeve të ndryshme të aplikacioneve softuerike, të cilat ekzekutohen në platforma të ndryshme dhe në sisteme operative të ndryshme. Arkitektura e ueb shërbimeve na tregon një koncept të përgjithshëm për kuptimin e ueb shërbimeve si dhe për marrëdhëniet ndërmjet komponentëve të këtij koncepti.

Komponentët kyçe të ueb shërbimeve janë: XML, SOAP, WSDL dhe UDDI.

XML – eXtensible Markup Language

Është gjuha më e përhapur dhe më e përdorur e cila njihet si gjuhë e përbashkët në llogaritjet biznesore.

Në botën reale sistemet e kompjuterëve dhe bazat e të dhënave përmbajnë të dhëna në formate të papapërshtatshme. Konvertimi i të dhënave me anë të XML-it e ul kompleksitetin e kohës dhe krijon të dhëna që mund të lexohen dhe kuptohen nga çfarëdo lloj aplikacioni. (ZENUNI, 2007)

XML gjuha markuese përshkruan strukturën dhe kuptimin e dokumentit, mirëpo nuk përshkruan formatizimin e elementeve në dokument. XML Dokumenti përmban vetëm tagjet që tregojnë përmbajtjen e dokumentit, mirëpo nuk tregon sesi duket ai dokument, ndërsa HTML bën formatimin e dokumentit. (ZENUNI, 2007)

Pjesë e XML kodit
<Student-PhD> <emri>Laurik</emri > <mbiemri>Helshani</mbiemri>

```
<universiteti>UET</ universiteti >
<vendi>Tiranë</vendi>
</ Student-PhD >
```

Tabela 4: Shembull nga kodi XML

Komponentët e XML-it

1. Document Type Definition (DTD)

Qëllimi i një DTD është për të përcaktuar strukturën e një XML dokument. DTD përcakton strukturën me një listë të elementeve dhe atributeve të lejuara brenda një XML-dokumenti. (XML DTD)

Shembulli i një XML-dokumenti me DTD të brendshme

```
<!DOCTYPE Student-PhD
[
<!ELEMENT Student-PhD (emri, mbiemri, universiteti, vendi)>
<!ELEMENT emri (#PCDATA)>
<!ELEMENT mbiemri (#PCDATA)>
<!ELEMENT universiteti (#PCDATA)>
<!ELEMENT vendi (#PCDATA)>
]>
<Student-PhD>
<emri>Laurik</emri >
<mbiemri>Helshani</mbiemri>
<universiteti>UET</ universiteti >
<vendi>Tiranë</vendi>
</ Student-PhD >
```

Tabela 5: Dokumenti XML me DTD

2. *XML Schema*

Një skemë XML-i përshkruan strukturën e një XML dokumenti, ashtu si një DTD. Një skemë XML-i shkruhet në XML, përkrah tipa të dhënash dhe namespaces. Grupe të

pavarura personash përmes XML-skemave mund të bien dakord në lidhje me një standard për shkëmbimin e të dhënave. (XML Schema)

Shembulli i një XML-skeme

```
<xs:element name="Student-PhD">

<xs:complexType>
  <xs:sequence>
    <xs:element name="emri" type="xs:string"/>
    <xs:element name="mbiemri" type="xs:string"/>
    <xs:element name="universiteti" type="xs:string"/>
    <xs:element name="vendi" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

Tabela 6: Skema e XML-it

3. *XML-Spaces*

Meqë brenda një XML-dokumenti mund të përmbahen edhe XML-dokumente të tjera, të cilat mund të definohen nga zhvillues të ndryshëm, ku për secilin nga ata, emri i njëjtë i një xml-atributi që i përdor në dokument mund të ketë kuptime të ndryshme. Konflikti mund të ndodhë po ashtu, nëse kombinojmë XML-dokumente nga aplikacione të ndryshme. Për të shmangur konflikte të tilla përdorim namespace të veçantë për secilin tag. (XML Namespaces)

SOAP

SOAP - Simple Object Access Protocol - është protokoll komunikimi që specifikon dhe e definon XML gjuhën për dërgimin dhe pranimin e mesazheve. SOAP shërben për komunikim ndërmjet aplikacioneve dhe është e dizajnuar për komunikim përgjatë internetit, po ashtu SOAP është platformë apo komponent i pavarur që bazohet në XML gjuhën. Qëllimi i SOAP-it është përshkrimi dhe formatimi i mesazhit që nuk është i lidhur me ndonjë arkitekturë harduerike apo softuerike, por me një që e bart mesazhin prej një platforme në ndonjë platforme tjetër. SOAP është XML i thjeshtë me protokoll të bazuar që bën lejimin dhe shkëmbimin e informacioneve përmes HTTP (potit 80), ose me thjeshtë mund të themi se SOAP është protokoll i cili na shërben për qasje të ueb shërbimit. SOAP do të zhvillohet si një W3C standard. SOAP siguron një rrugë të thjeshtë dhe një mekanizëm shumë të lehtë për shkëmbimin e informatave ndërmjet mjediseve të shpërndara duke përdorur XML. (XML Soap)

Struktura e SOAP-mesazhit

SOAP ka strukturë të thjeshtë. SOAP envelope përmban kokën (header) dhe trupin (body). Informatat e SOAP mesazhit që merren (nga RPC, XML ose error mesazhe) shkruhen në trupin e envelopes. Në kokë të SOAP mesazhit përmbahen informacione shtesë, p.sh informacione mbi nënshkrimet digjitale, informacione mbi transaksionet si dhe informacione mbi rrugëtimin. (ZENUNI, 2007)

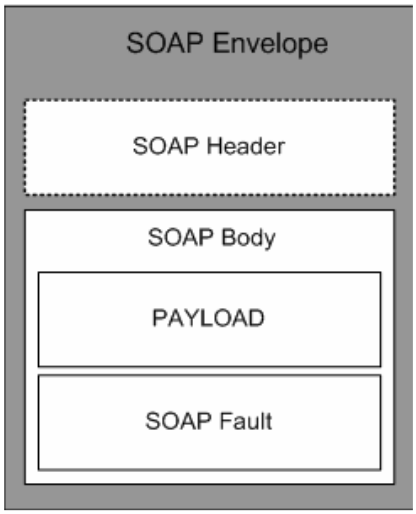


Figura 4: Struktura e SOAP-mesazhit

Skeleti i SOAP-mesazhit (XML Soap)
<?xml version="1.0"?> <soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope/" soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding"> <soap:Header> ... </soap:Header> <soap:Body> ... <soap:Fault> ... </soap:Fault> </soap:Body> </soap:Envelope>

Tabela 7: SOAP-Mesazhi

SOAP Envelope

SOAP Envelope është element i obligueshëm i cili paraqet elementin rrënje të një SOAP mesazhi. Ky element e definon XML dokumentin si një SOAP mesazh.

Çdo *SOAP envelope* duhet të përmbajë saktësisht një element *Body.* Elementi *Body* mund të përmbajë elemente fëmijë sipas nevojës, përmbajtja e *SOAP Body* - it është mesazh. Nëse *SOAP envelope* elementi në vete përmban SOAP header, atëherë maksimumi i headerave, që mund t'i ketë, është numri një (1) dhe duhet që SOAP headerin ta shfaqë si elementin fëmijë të parin më radhë, pra edhe para SOAP Body – it. Çdo element që përmbahet në SOAP Header quhet header bllok element. Qëllimi i një bllok headeri është që të komunikojë në përmbajtjen e informacioneve relevante për procedimin e SOAP mesazhit. (ZENUNI, 2007)

SOAP Header

SOAP heder nuk është element i obligueshëm për një SOAP mesazh. Ky element përmban informacionet e veçanta të aplikacionit (p.sh. si autentifikimin, nënshkrime digjitale, pagesat etj.) rreth SOAP mesazhit. Nëse elementi *SOAP header* është prezent në SOAP mesazh, atëherë ai duhet të jetë elementi i parë fëmijë i *SOAP envelope*. (Zenuni & Rexha, 2007)

SOAP Body

SOAP Body është element i detyrueshëm për një SOAP mesazh, ku në vete përmban mesazhet aktuale që janë vendimtare në fund të SOAP mesazhit.

Mesazhet përbrenda SOAP Body elementit duhet të kenë Namespaces. SOAP e definon një element përbrenda SAOP Body elementit me Namespace të nënkuptuar (ang. default)

(p.sh."http://www.w3.org/2001/12/soap-envelope"). Ky element është *SOAP Fault* elementi i cili përdoret për të treguar gabimet (ang. error) e mesazheve. Payload – Paraqet përmbajtjen e dokumentit që ne dëshirojmë ta dërgojmë. (ZENUNI, 2007)

SOAP Fault

SOAP Fault mesazhi është një mekanizëm i cili bën raportimin e gabimeve që ndodhin gjatë dërgimit të SOAP mesazhit. Kthimin e mesazhit si gabim e bën në nyjën e mëparshme të atij shtegu të dokumentit. Është detyrë e këtij seksioni për ta përcaktuar një mesazh komplet dhe me detaje, ku në të shpjegohet gabimi (SOAP Fault), kështu që në mënyrë të rastësishme mund të hasni në to në ueb shërbimin tuaj.

UDDI – Universal Discovery, Description Language

UDDI është platformë e pavarur dhe regjistër i cili bazohet në standardin e XML-it. UDDI iu shërben bizneseve që në mbarë botën të listojnë ueb shërbimet e tyre dhe të kërkojnë shërbime të tjera ose aplikacione softuerike që bashkëveprojnë me internetin. UDDI përshkruan tipin special të regjistrave, me anë të të cilëve bëhet listimi i ueb. UDDI siguron përshkrimin e biznesit dhe shërbimet e tij, zbulimin e ueb shërbimeve të tjera që i ofrojnë integrimin me biznese të tjera. (Zenuni & Rexha, 2007)

Tipat e regjistrave të UDDI-së

Regjistrat e UDDI-së mund të jenë të tipave të ndryshme:

Publik

Ky regjistër është i hapur për kërkime publike. Të gjitha hyrjet në regjistrin publik kopjohen edhe në regjistrat e tjerë. Nëse shfrytëzojmë një ueb shërbim publik, kjo na siguron një qasje të të gjitha shërbimeve të tjera. Shumë kompani të mëdha, përfshi këtu edhe IBM dhe Microsoft mbajnë regjistra të tillë publikë.

Privat

Ky regjistër qëndron prapa murit mbrojtës (firewall) të kompanisë. Qëllimi i këtij regjistri është kërkimi i shërbimeve të brendshme. Atij mund t'i qasen vetëm anëtarët e ndërmarrjes.

I kufizuar

Regjistrit të kufizuar mund t'i qasen vetëm disa kompani të caktuara, të cilave iu është dhënë e drejta e përdorimit të këtyre shërbimeve.

Informacioni në një regjistër ndahet në tri tipa, të cilat janë:

Faqet e bardha: Ato përmbajnë informacion bazë me adresa personale, emrat kontaktues dhe numra telefoni.

Faqet e verdha: Përmbajnë informacione për tipat e shërbimeve të biznesit që një kompani ofron.

Faqet e gjelbërta: Japin informacione teknike në lidhje me ueb shërbimet që janë të ekspozuara nga bizneset e ndryshme. P. sh..: përshkrimin e shërbimeve, rregullat e biznesit e të tjera. (Zenuni & Rexha, 2007)

Web Service Description Language (WSDL)

WSDL është sintaksë për të përshkruar ueb shërbimet, respektivisht mesazhet të cilat janë të shkëmbyera mes kërkuesit dhe ofruesit të ueb shërbimit. Një përshkrim i tillë jep një

vizion më të qartë të shërbimeve në aspektin e funksionimit të eksportimit e cila mund të thirret nga mesazhet hyrëse/dalëse.

WSDL nuk supozon se shkëmbimet mund të bëhen duke përdorur një formë të veçantë të komunikimit, dokumenti WSDL përcakton shërbime si koleksionet e rrjetit ose portet, pa marrë parasysh se çfarë formati kanë mesazhet ose protokollet e rrjetit që janë përdorur për të komunikuar. Portet bëjnë të mundur shkëmbimin e mesazheve, të cilat përcaktohen si përshkrime abstrakte të të dhënave të këmbimit. Përveç kësaj, WSDL përcakton një mekanizëm të përbashkët dhe të detyrueshëm. Kjo është përdorur për të bashkangjitur një protokoll të veçantë, format të të dhënave ose strukturë në një mesazh abstrakt, operacion, apo pikë të fundit. Figura e mëposhtme tregon terminologjinë WSDL duke treguar gjërat e ndryshme që WSDL përshkruan. Mesazhet e hyrjeve (inputs) dhe daljeve (outputs) formojnë një operacion dhe kombinimi i këtyre operacioneve formon një pikënisje. (Halili, 2014)

Versionet e WSDL-së

Deri më tani janë zhvilluar dy versione të WSDL-së:

- WSDL 1.1 është paraqitur nga kompani të mirënjohura si IBM, Ariba dhe Microsoft në mars të vitit 2001. Me këtë rast, këto kompani janë pajtuar që teknologjitë e mëparshme të tyre t'i bashkojnë për të krijuar një të vetme dhe më të fuqishme. SCL13 e Microsoft, Gjuha përshkruese e Shërbimeve dhe NASSL14 e IBM, të gjitha u zëvendësuan me WSDL.

- WSDL 2.0 ka trashëguar pothuajse të gjitha principet e WSDL 1.1, por është zgjeruar në shtresat e përshkrimit, aftësi të modularitetit dhe abstraksionit. (Halili, 2014)

Struktura e dokumentit të WSDL-së

Standardi i WSDL-së është interesant sepse e bën lejimin e cilitdo shërbim të rrjetës, jo vetëm të ueb shërbimeve. Çka do të thotë se pos SOAP mesazhit, cilido shërbim i rrjetës që kërkon formatimin e të dhënave për dërgim dhe kthim mund ta përdor WSDL-në.
WSDL dokumenti është XML dokument i cili përbëhet nga disa seksione kryesore të cilat ndahen në abstrakte dhe konkrete. (ZENUNI, 2007)

Elementet abstrakte përmbajnë pikat e fundme (end points) dhe mesazhet. Në abstrakt bëjnë pjesë:

- <wsdl: types>
- <wsdl: message>
- <wsdl: operation>
- <wsdl: portType>

Konkrete – Lidh pikat e fundme dhe mesazhet në një rrjetë reale dhe bën specifikimin e formatit të të dhënave. Në konkrete bëjnë pjesë këto elemente:

- <wsdl: binding>

- <wsdl: port>
- <wsdl: service> (ZENUNI, 2007)

Types përcakton tipin e të dhënave që përdorë ai ueb shërbim.

Message përshkruan përmbajtjet e mesazheve duke përdorur skemat XML.

PortType përcakton një operacion si një bashkësi mesazhesh që fillon dhe mbaron me ueb shërbimin.

Binding përcakton një format të veçantë për secilin *portType* element në WSDL.

Service Ky seksion na tregon aktualisht adresat e ndryshme që mund të përdoren për të komunikuar me këtë. (Zenuni & Rexha, 2007)

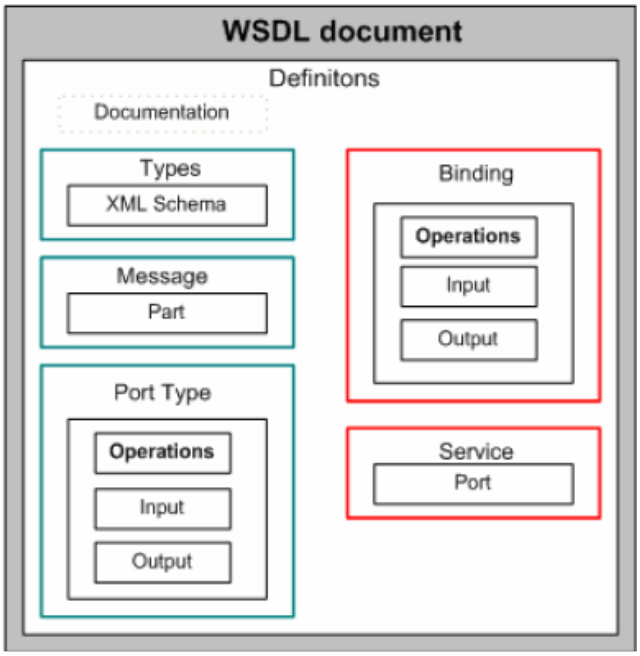


Figura 5: Struktura e WSDL-dokumentit

Shembull i një operacioni kërkesë-përgjigje në WSDL (XML WSDL)

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
     <soap:operation soapAction="http://example.com/getTerm"/>
     <input><soap:body use="literal"/></input>
     <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

Tabela 8: Kërkesë-përgjigje e një funksioni në WSDL

Representational State Transfer REST

Qasja REST është një arkitekturë e orientuar në burime (resource) për të ndërtuar sisteme të shpërndara duke përdorur ueb teknologji. Ky është një stil arkitektonik që trajton Ueb-in si një aplikacion të burimeve-qendrore. Për më tepër, çdo URL në një aplikacion REST përfaqëson një burim. REST është gjuha e aplikuar nga shumica e shërbimeve publike "interesante" të kohës dhe ka të ngjarë të përdoret në ekspozimin e gjërave sa më

interesante në të ardhmen. Në rast se ju publikoni të dhëna në mënyrën REST, ju jeni në një kompani të mirë. (Halili, 2014)

Roy Fielding ka arritur deri tek teknologjia REST duke vlerësuar të gjitha burimet e rrjeteve dhe teknologjive në dispozicion për krijimin e aplikacioneve të shpërndara. Pa ndonjë kufizim, çdo gjë përcillet, duke mundësuar zhvillimin e aplikacioneve që janë vështirë për t'u mirëmbajtur dhe zgjeruar. Duke pasur parasysh këtë, ai fillon me perceptimin e teknologjive që merren me resurset e shpërndara të aplikacioneve të stileve arkitekturore duke filluar me atë që ai e quan hapësira zero (e pavlefshme), që përfaqëson afinitetin e çdo teknologjie dhe të çdo zhvillimi të stileve të aplikacioneve pa rregulla dhe limite – dhe përfundon me kufizimet e me resurset e shpërndara të aplikacioneve të stileve arkitekturore duke filluar me atë që ai e quan hapësira zero (e pavlefshme), e cila përfaqëson afinitetin e çdo teknologjie dhe të çdo zhvillimi të stileve të aplikacioneve pa rregulla dhe limite; përfundon me kufizimet e mëposhtme që definojnë sistemin RESTful.

- duhet të jetë një sistem klient-server,
- duhet të jetë pa gjendje – që nënkupton se nuk ka nevojë që shërbimi të mbajë sesione për përdoruesit, në mënyrë më specifike, çdo kërkesë duhet të jetë e pavarur nga të tjerat,
- duhet të mbështesë një sistem deponimi – infrastruktura e rrjetit duhet të mbështesë depot në nivele të ndryshme,
- duhet të jetë në mënyrë të njëjtë e qasshme - çdo burim duhet të ketë një adresë unike dhe një pikë të vlefshme të qasjes,

- Duhet të jetë i shtresëzuar - duhet të mbështesë shkallëzimin. Ajo duhet të sigurojë kod mbi kërkesën - edhe pse kjo është një kufizim opcional, aplikacionet mund të jenë shtruar në kohën e duhur duke lejuar shkarkimin e kodit të kërkesës.

Atributet e mësipërme nuk na këshillojnë ne se çfarë lloj të teknologjisë duhet të përdorim, por ato specifikojnë se si të dhënat janë të transmetuara midis komponentëve dhe cilat janë avantazhet dhe udhëzimet e REST. Nuk është e nevojshme shpikja e teknologjive të reja ose e protokolleve të rrjeteve, por ne mund të përdorim infrastrukturën ekzistuese të rrjeteve siç është rasti kur uebi krijon arkitekturat RESTful. (Halili, 2014)

Foljet REST

- GET - përdoret për të marrë informacione, që do të thotë se shfletuesi vë GET në disa URI dhe rinxjerr një përfaqësim si për shembull: një HTML, (tekst të thjeshtë, JPG, etj) të burimeve të caktuara nga ai URI.
- POST - është përdorur për të futur informacion të ri, ndërkohë duke treguar lidhjen e saj me informacionin e vjetër.
- PUT - për azhurnimin e informacionit.
- DELETE - të fshini informacionin.

Është e rëndësishme të përmendet se emrat tanë nuk janë universalë dhe foljet nuk janë polimorfive, që do të thotë se duke përdorur folje të ndryshme për çdo emër i gjithë komunikimi mund të bëhej i pamundur. (Halili, 2014)

Hyper Text Transfer Protocol (HTTP)

Është protokolli më i popullarizuar dhe më i përdoruri në Internet. HTTP kryesisht është krijuar në fillimin e viteve 1990. Ky protokoll është krijuar për t'i ndihmuar shkencëtarët për të bërë gjetjen dhe ndarjen e informatave duke iu lejuar që përmes disa linçeve të thjeshta të bëhet kalimi nga një dokument në dokumentin tjetër. Përdorimi dhe zhvillimi i HTTP-së është rritur me një vrull shumë të shpejtë. Kështu që ueb sajtet përdorin HTTP-në si mekanizëm komunikimi, por kjo nuk do të thotë se HTTP është e kufizuar vetëm në WWW, mirëpo në fakt ueb-i është vetëm një aplikacion që përdor HTTP-në për të bartur informacionet ndërmjet serverit dhe klientit. (ZENUNI, 2007)

HTTP lidhja krijohet me anë të një pajtimi të thjeshtë mes klientit dhe serverit (Ang.: handshaking). Më rëndësi të cekët është se inicimin për krijimin e kësaj lidhje mes Klientit/Serverit e bën kompjuteri (PC) i klientit. Klienti kërkesat i bën me anë të HTTP protokollit. (ZENUNI, 2007)

HTTP është një standard që paraprin hyrjen e ueb shërbimeve, i cili është zhvilluar për ta lehtësuar transferimin e kërkesave prej brouserit në ueb server. Trafiku i HTTP –së është jashtëzakonisht i mirë, shumica e mureve mbrojtëse e lejojnë trafikun pa ndonjë problem dhe pa ndonjë konfiguracion të veçantë.

Shkëmbimi i SOAP mesazheve dhe WSDL dokumenteve prej një kompjuteri në kompjuterin tjetër bëhet përmes HTTP protokollit (portit 80). Deri më tani shumica e Ueb shërbimeve janë ndërtuar me HTTP. (ZENUNI, 2007)

HTTP komunikon përmes TCP/IP protokollit. (ZENUNI, 2007)

3.4. Ueb shërbimet e Google-it

Pasqyrë e përgjithshme

Janë një përmbledhje e ndërfaqeve HTTP që ofrojnë të dhëna gjeografike për aplikacionet që ndërlidhen me harta. (Web Services)

Këtu unë do të bëj një hyrje në lidhje me disa nga ato që kam përdorur për arritjen e disa prej objektivave të punimit praktik doktoral.

Këto shërbime ofrohen si RESTful. Pra, përdoren duke bërë një kërkesë HTTP në një URL të veçantë, ndërsa parametrat (obligativë dhe joobligativë) ia kalojmë si argumente në URL. Zakonisht rezultati i HTTP-kërkesës kthehet si JSON ose XML, për t'u përpunuar më tutje nga aplikacioni ku integrohen këto shërbime. (Web Services)

Saktësia e formatimit të përgjigjes së ueb shërbimit në lidhje me HTTP-kërkesën nuk është e garantuar, meqë ndodh që disa elemente mungojnë në një apo më shumë vende. Nuk duhet supozuar që forma e caktuar e përgjigjes do të kthehet për kërkesa të ndryshme. Në vend të kësaj duhet që përgjigjja të përpunohet dhe përmes query-ive të ndryshëm të përzgjidhen vlerat e duhura. (Web Services)

Ueb shërbimet e Google-it ofrojnë përgjigje lehtë të kuptueshme, mirëpo nuk janë shumë miqësore (not user friendly) për përdoruesin. Pra, është e nevojshme që të kuptohet përgjigjja e kthyer dhe me anë të ndonjë gjuhe programuese të filtrohen vetëm vlerat që janë me rëndësi jetike për projektin (qoftë gjatë shfaqjes apo për përpunim të mëtejshëm). (Web Services)

Ueb shërbimet e Google-it më të njohurat për programuesit janë: (Web Services)

- Google Maps Distance Matrix API.
- Google Maps Directions API.
- Google Maps Geocoding API.
- Google Maps Geolocation API.
- Google Places API Web Service.
- Google Maps API.

Google Maps Distance Matrix API

Google Maps Distance Matrix API është një shërbim i cili ofron distancën dhe kohën e udhëtimit nga një pikë e dhënë deri te destinacioni i dëshiruar. Rezultati i kthyer është i bazuar në rrugën e rekomanduar mes pikës fillestare dhe asaj ku synojmë të shkojmë, llogaritur nga Google Maps API, dhe përbëhet nga rreshta të cilët përmbajnë distancën dhe kohëzgjatjen e udhëtimit mes çdo palë pikave të dhëna nga A deri te pika B. Pra, ky shërbim mund të përdoret për gjetjen e distancës mes dy apo më shumë pikave të dhëna si çifte. (Distance Matrix API)

Ky shërbim nuk kthen informata të detajuara në lidhje me rrugën. Informatat e detajuara në lidhje me rrugën mund t'i marrim duke ia kaluar ueb shërbimit: Google Maps Directions API një çift të vetëm parametrash: ngaKu deriKu dëshirojmë të udhëtojmë. (Distance Matrix API)

Shpjegimi i mëposhtëm ka për qëllim zhvilluesit, të cilët dëshirojnë për të llogaritur distancën e udhëtimit dhe kohën mes një numri të pikave duke përdorur Google Maps Distance Matrix API. (Distance Matrix API)

Distance Matrix – kërkesa

Ka formën e mëposhtme: (Distance Matrix API)

https://maps.googleapis.com/maps/api/distancematrix/*output*?*parameters*

https rekomandohet për aplikacionet që përfshijnë të dhëna të ndjeshme të përdoruesve, të tilla si vendndodhja e një përdoruesi etj.

Përmes fjalës çelës ***output,*** e cila mund t'i marrë vlerat JSON ose XML, i tregojmë këtij shërbimi të Google-it, që rezultatin e dëshirojmë të formatuar në JSON apo XML.

Parametrat e kërkesës

Disa nga parametrat janë obligativë, ndërsa të tjerët janë opcionalë. Çdo parametër duhet ndarë nga njëri- tjetri sipas standardit të URL-së duke përdorur karakterin ***&***.

Parametrat obligativë (Distance Matrix API)

origins – Pika e fillimit për llogaritjen e distancës së udhëtimit dhe kohës. Mund të futen për llogaritje një apo më shumë vende të ndara nga njëra-tjetra përmes shenjës "|", në formë të një adrese, gjerësisë/gjatësisë gjeografike apo Id-së së një vendi të Google-it. P. sh.:

origins=41.329430, 19.820226 ose

origins=Tiranë ose

origins=ChIJN1t_tDeuEmsRUsoyG83frY4

destinations – Një apo më shumë vende mund të përdoren si pikësynim për të llogaritur distancën dhe kohën e udhëtimit. Forma e parametrave është e njëjtë si e atyre të ***origins***.

Parametrat joobligativë (opcionalë) (Distance Matrix API)

mode - Përcakton mënyrën e transportit të përdorur kur bëhet llogaritja e distancës. Detaje të tjera do të sqarohen në vazhdim.

- *bicycling* – llogarit distancën e rrugës së biçikletave (nëse është në dispozicion),
- *driving* - llogarit distancën e rrugës duke përdorur rrjetin rrugor,
- *walking*- llogarit distancën e rrugës për të ecur duke përdorur shtigjet dhe trotuaret. (Distance Matrix API)

Sistemi për njësi matëse

units=metric – kthen rezultatin në kilometra dhe metra,

units=imperial – kthen rezultatin në miles dhe feed.

Shembull

https://maps.googleapis.com/maps/api/distancematrix/json?origins=41.329430, 19.820226

&destinations=41.324886, 19.469565&key=API_KEY

Përgjigjja si JSON


```
{
  "destination_addresses" : [ "Rruga Rinia Durrsake, Durrës, Albania" ],
  "origin_addresses" : [ "Rruge Devijimi, Tiranë 1001, Albania" ],
  "rows" : [
    {
      "elements" : [
        {
          "distance" : {
            "text" : "35.2 km",
            "value" : 35222
          },
          "duration" : {
            "text" : "33 mins",
```

```
            "value" : 2000
          },
          "status" : "OK"
        }
      ]
    }
  ],
  "status" : "OK"
}
```


Tabela 9: Përgjigjja e ueb shërbimit Distance Matrix si JSON

https://maps.googleapis.com/maps/api/distancematrix/xml?origins=41.329430,19.820226&destinations=41.324886,19.469565&key=API_KEY

Përgjigja si XML


```
<?xml version="1.0" encoding="UTF-8"?>
<DistanceMatrixResponse>
	<status>OK</status>
	<origin_address>Rruge Devijimi, Tiranë 1001, Albania</origin_address>
	<destination_address>Rruga Rinia Durrsake, Durrës, Albania</destination_address>
	<row>
		<element>
			<status>OK</status>
			<duration>
				<value>2000</value>
				<text>33 mins</text>
			</duration>
			<distance>
				<value>35222</value>
				<text>35.2 km</text>
			</distance>
		</element>
	</row>
</DistanceMatrixResponse>
```


Tabela 10: Përgjigjja e ueb shërbimit Distance Matrix si XML

Google Maps Distance Matrix – Përgjigjja

Përgjigjja e Google Maps Distance Matrix përmban elementet e mëposhtme:

status- fusha e statusit brenda objektit të përgjigjes përmban statusin e kërkesës dhe mund të përmbajë informata të dobishme për debugging.

origin_addresses- përmban një grup adresash të formatizuar nga geocoder dhe të lokalizuara në bazë të gjuhës së dhënë si parametër te kërkesa.

destination_addresses – e njëjtë ashtu si tek origin_addresses me të vetmin dallim që këto formatizohen vetëm nëse një gjë e tillë është e përshtatshme.

rows - përmban një grup të elementeve të cilat përsëri përmbajnë në vetvete statusin, kohën dhe distancën udhëtimit. (Distance Matrix API)

Kufizimet në përdorimin e shërbimit

Përdoruesit standard pa pagesë

1) 2 500 query brenda ditës.

2) 100 elemente brenda një query.

3) 100 elemente për 10 sekonda.

4) Për çdo 1000 elemente shtesë, duhet paguar 0.5$ dhe kjo mund të vazhdohet deri në 100000 elemente shtesë brenda 24 orë.

Përdoruesit premium

1) Kuota e përditshmërisë është 100 000 elemente brenda 24 orë, kërkesat shtesë zbatohen në vlerën e kredive të blera për vit për API.

2) 625 elemente për query.

3) Përpunohen 1000 elemente për sekondë.

4) Për klientët me plan premium ka përkrahje teknike të Google-it 24 orë/7 ditë të javës. (Distance Matrix API)

Google Maps Directions API

Është shërbim i cili llogarit drejtimet në mes të lokacioneve duke përdorur një HTTP-kërkesë. Mund të kërkohet për drejtimet për disa mënyra të transportit, duke përfshirë transit, vozitje, në këmbë ose çiklizëm. Te ky shërbim parametrat si origjina (pika fillestare), destinacioni dhe pikat e rrugës mund të përcaktohen si tekst apo si koordinata GPS-i (gjerësi gjeografike/gjatësi gjeografike). (Directions API)

Ky shërbim është i dizajnuar në përgjithësi për llogaritjen e drejtimeve në lidhje me adresat statike (të ditura që më parë) dhe për aplikacionet që përdorin harta. Pra, ky shërbim nuk është projektuar për t'iu përgjigjur në kohë reale të dhënave që jepen aty për aty.

Llogaritja e drejtimeve shpenzon kohë dhe burime dhe për këtë arsye nëse është e mundur rezultatet ruhen përkohësisht në dizajnin e aplikacioni ku përdoret ky shërbim. (Directions API)

Kërkesa e Google Maps Directions

Ka formën e mëposhtme:

https://maps.googleapis.com/maps/api/directions/*output*?*parameters*

https rekomandohet për aplikacionet që përfshijnë të dhëna të ndjeshme të përdoruesve, të tilla si vendndodhja e një përdoruesi etj.

Fjalës çelës output mund t'i bashkëngjisim vlerën JSON ose XML dhe përmes këtyre ne i tregojmë këtij shërbimi se në çfarë formati e dëshirojmë përgjigjen. (Directions API)

Parametrat obligativë

origins – Pika e fillimit për llogaritjen e distancës së udhëtimit dhe kohës. Mund të futën për llogaritje një apo më shumë vende të ndara nga njëra-tjetra përmes shenjës "|", në formë të një adrese, gjerësisë/gjatësisë gjeografike apo Id-së së një vendi të Google-it. P. sh.:

origins=41.329430, 19.820226 ose

origins=Tiranë ose

origins=ChIJN1t_tDeuEmsRUsoyG83frY4

destinations – Një ose më shumë vende mund të përdoren si pikësynim për të llogaritur distancën dhe kohën e udhëtimit. Forma e parametrave është e njëjtë si e atyre të ***origins***.

Parametrat joobligativë

mode - Përcakton mënyrën e transportit të përdorur kur bëhet llogaritja e distancës. Detaje të tjera do të sqarohen në vazhdim. (Directions API)

- *bicycling* – llogarit distancën e rrugës së biçikletave (nëse është në dispozicion),
- *driving* - llogarit distancën e rrugës duke përdorur rrjetin rrugor,
- *walking*- llogarit distancën e rrugës për të ecur duke përdorur shtigjet dhe trotuaret.

language- Specifikon gjuhën në të cilën të kthehen rezultatet. Nëse nuk është e dhënë gjuha, atëherë shërbimi mundohet të aplikojë gjuhën amtare të domain-it nga i cili është dërguar kërkesa. Lista e gjuhëve në dispozicion është:

Language Code	Language	Language Code	Language
Ar	Arabic	Kn	Kannada
Bg	Bulgarian	Ko	Korean
Bn	Bengali	Lt	Lithuanian
Ca	Catalan	Lv	Latvian
Cs	Czech	Ml	Malayalam
Da	Danish	Mr	Marathi
De	German	Nl	Dutch
El	Greek	No	Norwegian
En	English	Pl	Polish
en-AU	English (Australian)	Pt	Portuguese
en-GB	English (Great Britain)	pt-BR	Portuguese (Brazil)
Es	Spanish	pt-PT	Portuguese (Portugal)
Eu	Basque	Ro	Romanian
Eu	Basque	Ru	Russian
Fa	Farsi	Sk	Slovak
Fi	Finnish	Sl	Slovenian
Fil	Filipino	Sr	Serbian
Fr	French	Sv	Swedish
Gl	Galician	Ta	Tamil
Gu	Gujarati	Te	Telugu
Hi	Hindi	Th	Thai
Hr	Croatian	Tl	Tagalog
Hu	Hungarian	Tr	Turkish
Id	Indonesian	Uk	Ukrainian
It	Italian	Vi	Vietnamese
Iw	Hebrew	zh-CN	Chinese (Simplified)
Ja	Japanese	zh-TW	Chinese (Traditional)

Tabela 11: Lista e gjuhëve në dispozicion të përgjigjes së shërbimeve

alternatives- Nëse vendoset alternatives=true, atëherë serveri kthen një përgjigje ku përmbahen më tepër se një rrugë (edhe disa rrugë të tjera alternative shtesë), mirëpo duhet pasur parasysh se kjo ndikon në zgjatjen e kohës së përgjigjes prej serverit. (Directions API)

Përgjigja e Google Maps Direction API

Status Code

status – kjo fushë përmban informata që i shërbejnë përdoruesit të shërbimit për debugging, të cilat tregojnë shkakun e dështimit të shërbimit. Fusha *status* mund të përmbajë vlerat e mëposhtme:

- OK - tregon që përgjigjja nga serveri përmban një rezultat të vlefshëm.
- NOT_FOUND - tregon që të paktën një nga vendet e specifikuara në origjinën e kërkesës, destinacion, apo waypoints, nuk mund të geocoded (procesi i kthimit nga adresa ose emri i dhënë i ndonjë vendi në koordinatat GPS, lat/lng).
- MAX_WAYPOINTS_EXCEEDED - tregon se janë dhënë shumë waypoints në kërkesë. Për aplikimet që përdorin Google Maps API Directions si një shërbim ueb-i, numri maksimal i lejuar i waypoints është 23, plus origjina dhe destinacioni.
- INVALID_REQUEST - tregon se kërkesa e dhënë ishte e pavlefshme, ndërsa shkaqet mund të jenë që : ndonjë parametër ose ndonjë vlerë e parametrit mund të jenë dhënë gabim.

- OVER_QUERY_LIMIT - tregon që shërbimi është përdorur më shumë sesa është i lejuar brenda një periudhe të caktuar kohore, dhe kjo është e lidhur me statusin e përdoruesit nëse është përdorues standard apo premium i shërbimeve të Google-it.
- REQUEST_DENIED - tregon që kërkesa e bërë nga aplikacioni, për të shfrytëzuar këtë shërbim të Google-it, është refuzuar nga serveri i Google-it.
- UNKNOWN_ERROR - tregon që kërkesa nuk mund të përpunohet nga serveri për shkak të një gabimi të panjohur, dhe mund të ketë sukses nëse riprovohet.

(Directions API)

Kufizimi në përdorim

Përdoruesit standardë

1) 2 500 kërkesa falas brenda ditës.
2) Deri në 23 waypoints të lejuara për çdo kërkesë, që përmban një API_KEY, ose deri në 8 waypoints nëse kërkesa nuk përmban API_KEY.
3) 10 kërkesa brenda sekondit.
4) Për çdo 1000 kërkesa shtesë duhet paguar 0.5 $ dhe kjo mund të vazhdojë kështu deri në 100 000 kërkesa brenda ditës.

(Directions API)

Përdoruesit Premium

1) Iu ndahen kuota të përditshme, 100 000 kërkesa për 24 orë;

2) Kërkesat shtesë të aplikuara zbatohen në kuadër të blerjes vjetor të Maps API- kredive.

3) Përpunohen 50 kërkesa brenda sekondit.

4) Kanë 24 orë / 7 ditë të javës përkrahje teknike.

(Directions API)

Google Maps API

Google Maps API lejon integrimin e hartave të Google-it në ueb aplikacione të ndryshme. Zhvilluesit këtë e bëjnë duke përdorur ndërfaqe të JavaScript ose Flash-it. Ky shërbim është dizanjuar të punojë po ashtu në windows-aplikacione si dhe në pajisje mobile. API përfshin lokalizimin e më shumë se 50 gjuhëve, lokalizimin e rajoneve të ndryshme dhe geocoding, dhe ka mekanizma për zhvilluesit (programuesit) që zhvillojnë aplikacione për ndërmarrjet, por dhe që duan të përdorin API Google Maps brenda një intraneti. Ky shërbim është i besueshëm meqë zhvilluesi (programuesi) i qaset përmes secured HTTP (https). (Google Maps for every platform)

Për pajisjet mobile me sistem operativ IOS, API adekuat është Google Places API for IOS, ndërsa për pajisjet me sistem operativ Android, API adekuat është: Google Maps Android API. Përmes këtyre API-ve na mundësohet integrimi i këtyre gjërave në aplikacionet e pajisjeve mobile: integrimi i hartave bazë të Google-it, i ndërtesave 3D, pamje të rrugëve dhe imazhe satelitore, shënjues të vendeve të ndryshme etj. (Google Maps for every platform)

Google Maps përveçse siguron një ndërfaqe intuitive dhe shumë të përgjegjshme të hartës me imazhet satelitore dhe të dhëna të detajuara në lidhje me rrugët, i mundëson zhvilluesit që të shtojnë Overlayers mbi të, në mënyrë që përdoruesi ta ketë në kontroll të plotë gjatë

navigimit. Pra, thënë shkurt, përmes këtij API mundësohet personalizimi i hartave të Google-it sipas nevojave specifike të aplikacionit. (Google Maps for every platform)

Google Maps Android API

Me API Google Maps Android, ju mund të shtoni harta të bazuara në të dhënat e Google Maps në aplikacionin që zhvilloni. API automatikisht merret me qasjen në serverat e Google Maps, me shkarkimin e të dhënave, paraqitjen e hartës dhe me ndërveprimin hartë - përdorues. Ju gjithashtu mund të përdorni thirrje API për të shtuar shënjues, poligone dhe overlays mbi hartën bazë, për të ndryshuar pamjen e përdoruesit të një zone të caktuar në hartë. Këto objekte ofrojnë informata shtesë në lidhje me ndonjë vend në hartë, por dhe lejojnë ndërveprimin e përdoruesit me hartë. (Android)

Overlayers (shtresat e vendosura mbi hartë Google)

Janë objekte që vendosen mbi hartë të Google-it dhe janë të lidhura me të përmes koordinatave gjeografike. Harta e Google-it ofron disa lloje prej tyre, mirëpo unë po cek vetëm ato që kam përdorur në aplikacionin për të arritur objektivat e punës praktike të doktoratës sime.

Markers – shërbejnë për të shënjuar vende të caktuara në hartë. Në aplikacionin tim përdoruesi shënjon vendet që dëshiron ai t'i vizitojë.

Polyline - Seri e linjave të drejta në një hartë. Përmes tyre në aplikacionin e androidit vizatoj rrugën më të shkurtër që lidh të gjitha vendet për t'u vizituar, pra ato vende që përdoruesi dëshiron t'i vizitojë.

Info-Window – përmes tyre e informoj përdoruesin me detaje më të hollësishme në lidhje me vendin e zgjedhur për vizitë. Këto janë pjesë e pandashme e shënjuesve (markers). (Android)

The Google Maps Geocoding API

Geocoding është procesi i konvertimit të adresave si: *Rexhep Krasniqi, Prishtinë, Kosovë* në koordinata gjeografike (gjerësi 42.644353 dhe gjatësi 21.150527 gjeografike) të cilat mund të përdoren për vendosjen e shënjuesve (markers) në hartë të Google-it.

Reverse Geocoding është procesi i kundërt i kthimit nga koordinatat gjeografike në një adresë të lexueshme. (Web Services)

Google Maps Geocoding API ofron një mënyrë të drejtpërdrejtë për të hyrë në këto shërbime nëpërmjet një kërkesë HTTP. Shërbimi është menduar për zhvillues të ueb-aplikacioneve dhe për zhvillues të aplikacioneve mobile që dëshirojnë të përdorin të dhënat geocoding në hartat e ofruara nga një prej Google Maps API. (Web Services)

Shembull:

https://maps.googleapis.com/maps/api/geocode/json?&address=Bulevardi Gjergj Fishta, Tirana, Albania

Përgjigjja në JSON e shërbimit të kërkuar

```
{
  "results" : [
    {
      "address_components" : [
        {
          "long_name" : "Bulevardi Gjergj Fishta",
          "short_name" : "Bulevardi Gjergj Fishta",
          "types" : [ "route" ]
```

```
      },
      {
        "long_name" : "Tirana",
        "short_name" : "Tirana",
        "types" : [ "locality", "political" ]
      },
      {
        "long_name" : "Tiranë",
        "short_name" : "Tiranë",
        "types" : [ "administrative_area_level_2", "political" ]
      },
      {
        "long_name" : "Tirana",
        "short_name" : "Tirana",
        "types" : [ "administrative_area_level_1", "political" ]
      },
      {
        "long_name" : "Albania",
        "short_name" : "AL",
        "types" : [ "country", "political" ]
      }
    ],
    "formatted_address" : "Bulevardi Gjergj Fishta, Tirana, Albania",
    "geometry" : {
      "bounds" : {
        "northeast" : {
          "lat" : 41.3236845,
          "lng" : 19.8174208
        },
        "southwest" : {
          "lat" : 41.3205554,
          "lng" : 19.7921714
        }
      },
      "location" : {
        "lat" : 41.321943,
        "lng" : 19.8052316
      },
      "location_type" : "GEOMETRIC_CENTER",
      "viewport" : {
        "northeast" : {
          "lat" : 41.3236845,
          "lng" : 19.8174208
        },
```

```
            "southwest" : {
              "lat" : 41.3205554,
              "lng" : 19.7921714
            }
          }
        },
        "place_id" : "ChIJJ6fuiKoxUBMRRNUV9AX2gic",
        "types" : [ "route" ]
      }
   ],
   "status" : "OK"
}
```

Tabela 12: Përgjigjja e ueb shërbimit geo-code në JSON

(Web Services)

Google Places API

Të dhënat lokale për mbarë botën. Të dhënat thirren nga e njëjta bazë e dhënash të cilën e shfrytëzojnë Google Maps dhe Google+. Përmes këtij shërbimi përdoruesi qaset në më tepër se një milion biznese, pika interesi dhe bukuri natyrore, bazë e të dhënave e cila është përditësuar nga përdorues të moderuar dhe vërtetuar nga poseduesit e këtyre bizneseve. (Places API for Android)

Google Places API për Android

Për të përdorur Google Places API për Android, duhet të shtohet një Google API kyç në aplikacionin ku integrohet ky ueb shërbim. Lloji i çelësit të API-t që duhet shtuar është një çelës i Android-it.

Të gjitha Aplikacionet në Android janë nënshkruar me një certifikatë dixhitale, ku për secilën nga to zhvilluesi i tyre mban një çelës privat. Çdo çelës i ndonjë API të caktuar

është i lidhur ngusht me një certifikatë të veçantë pa marrë parasysh sesa përdorues e përdorin atë aplikacion. (Places API for Android)

Te konsola e zhvilluesve të Google-it (Google Developers Console) mund të merret një çelës i tillë, ku duhet të kyçemi përmes llogarisë së email-it të Google-it. Në këtë rast duhet aplikuar për një Çelës Androidi e jo për një Çelës Brouseri. Po i njëjti çelës mund të përdoret më tutje edhe për Google Maps API për Android. (Places API for Android)

Përmes këtij shërbimi bëhet implementimi i njohjes së vendit përmes pajisjes android, propozimi dhe kompletimi automatik i vendeve gjatë kërkimit në bazë të një mostre të caktuar si dhe shtimi i informacioneve në aplikacionin e zhvilluar në lidhje me miliona vende të ndryshme. (Android)

Kompletimi dhe propozimi automatik gjatë kërkimit

Ueb shërbimi i Google-it në linkun e mëposhtëm e mundëson zhvilluesin e aplikacionit të androidit që ta pajisë aplikacionin e tij me një funksion shumë të avancuar dhe shumë atraktiv për përdoruesin, pikërisht kërkimi i ndonjë vendi në bazën e të dhënave të Google-it në bazë të një mostre të caktuar (qoftë ajo pjesë e emrit apo e adresës). (Android)

https://maps.googleapis.com/maps/api/place/autocomplete/*output*?*parameters*

Serveri i Google-it përgjigjet me një listë vendesh, emrat apo adresat e të cilëve i përshtaten mostrës për kërkim, në formatin tashmë të njohur si XML ose JSON. (Android)

Detajet e një vendi të caktuar

Ueb shërbimi i Google-it në linkun e mëposhtëm e mundëson zhvilluesin e aplikacionit të androidit që ta pajisë aplikacionin e tij me një funksion shumë të avancuar dhe shumë atraktiv për përdoruesin, pikërisht nxjerrja e informacioneve të detajizuara në lidhje me vendin e prekur (klikuar) në hartën e Google-it që është e integruar në aplikacion. (Android)

https://maps.googleapis.com/maps/api/place/details/*output*?*parameters*

Përmes eventit: 'onMapClick' në hartën e integruar të Google-it në aplikacionin e androidit nxirren koordinatat gjeografike të atij vendi dhe përmes shërbimit në linkun e mësipërm dërgohet kërkesa në server të Google-it. Serveri i Google-it përgjigjet me një listë informacionesh në lidhje me atë vend, siç janë: emri, adresa, id-ja unike e atij vendi në server të Google-it, koordinatat gjeografike, nën administrimin e kujt është ai vend etj. (Android)

Krahasimi mes Google Maps dhe Bing Maps

Google Maps

- Falas për përdorim të ueb faqeve publike (d. m. th.: ueb faqe, për të cilën nuk duhet të paguhet për qasje në harta). Ekziston një kufizim në numrin se sa herë ngarkohet

harta e Google-it në një ueb faqe të caktuar, mirëpo ky numër nuk është publikuar në faqen zyrtare të Google-it.

- Mund të përdoret për përdorim komercial (nëse një ueb faqe e caktuar është fitimprurëse).
- Reklama pa pagesë mund të shfaqën në hartën tuaj (me paralajmërim 90 ditë përpara).
- Falas në kuptimin e të dhënave që shfaq ndokush në hartat e Google-it, atëherë Google-i ka të drejtë që ato t'i përdorë për marketing. Kjo është e definuar te kushtet e përdorimit që personi në këtë mënyrë i jep Google-it të drejtë të përjetshme, të pakthyeshme, në të gjithë botën, pa paguar ndonjë taksë, dhe licencë joekskluzive për të riprodhuar, përshtatur, modifikuar që ta publikojë përmbajtjen tuaj si rezultatet e kërkimit përmes ueb shërbimit për kërkim.
- Pa marrëveshje në nivel shërbimi - në qoftë se ai shkon offline ju nuk keni replikë.
- Kushtet e marrëveshjes mund të ndryshojë në çdo kohë pa paralajmërim, që do të thotë se një ditë prej ditësh mund të mos jenë falas.
- Ju jep qasje në të dy API-it: ajax-Mapping-API dhe në API-n e ueb shërbimeve të lidhura me hartat.
- Nuk mund të përdoret për navigim në kohë reale nga persona të palës së tretë, për shkak të marrëveshjes në lidhje me palën e tretë.
- Mund të përdoret për përcjelljen e aseteve, por vetëm për shërbimin i cili i ofrohet falas përdoruesit fundor.

(Marshall, 2010)

Bing Maps

- Falas për përdorim të ueb faqeve publike, ku numri i përdorimit është i kufizuar deri në 125 mijë ngarkesa të hartës në vit ose 500 000 transaksione në hartë (ku një transaksion nënkupton 8 pllaka katrore harte 256-pixel apo një kërkim adrese) në vit.
- Përdorim falas për organizatë jofitimprurëse apo organizatë arsimore (ku rekomandohet shikimi i termave dhe kushteve për të siguruar që organizata kualifikohet si e tillë)
- Mund të përdoret për përdorim komercial (nëse një ueb faqe e caktuar është fitimprurëse)
- Pa marrëveshje në nivel shërbimi - në qoftë se ai shkon offline ju nuk keni replikë
- Kushtet e marrëveshjes mund të ndryshojë në çdo kohë pa paralajmërim, do të thotë që një ditë prej ditësh mund të mos jenë falas
- Ju jep qasje në të dy API-it: Ajax-Mapping-API dhe në API-n e ueb shërbimeve të lidhura me hartat.
- Gjitha të dhënat që ju në Bing-Maps dërgoni, i jepni Microsoft-it një licencë për ripërdorimin e tyre, mirëpo jo për të dhënat personale apo të një personi të tretë. Pra, kjo vlen vetëm për të dhënat që janë hostuar në emër të Microsoft-it.
- Pa pagesë në kuptimin që ndonjë reklamë mund të paraqitet në ndonjë pikë të hartës në të ardhmen
- Nuk mund të përdoret për përcjelljen e aseteve pa licencë komerciale

- Nuk mund të përdoret për navigim në kohë reale nga persona të palës së tretë, për shkak të marrëveshjes në lidhje me palën e tretë.

(Marshall, 2010)

Licenca komerciale

Nëse ju doni të përdorni hartat më shpesh sesa ofrohet nga licencimi falas, ose dëshironi të përdorni një tipar jo në dispozicion në bazë të licencës falas ju do të duhet të paguani për një licencë. Edhe pse ne nuk mund të japim detaje çmimi për API, por mund të krahasojmë disa karakteristika dhe të komentojmë në përgjithësi në lidhje sesi funksionon licencimi.
Një pyetje që parashtrohet shpesh është se: a përfitohen imazhe më cilësore të hartave apo shërbim më i shpejtë kundrejt një pagese të licencës komerciale? Përgjigjja për këtë pyetje dhe që vlen për të dyja hartat është JO. Edhe pse paguhet licenca komerciale përdoret i njëjti shërbim. (Marshall, 2010)

Google Maps

- Është e mbuluar nga një marrëveshje në nivel shërbimi, por jo dhe për shërbimet geocoding etj.
- Është në dispozicion për një çmim fiks me një numër të caktuar thirrjesh apo ngarkimi të këtyre hartave.
- Është në dispozicion për përdorim në ueb faqe private.
- Garanton që nuk fusin reklama apo nuk mund të paraqiten reklama pa lejen tuaj.

(Marshall, 2010)

Bing Maps

- Është e mbuluar nga një marrëveshje në nivel shërbimi për të gjitha shërbimet.
- Shumë mënyra licencimi, duke përfshirë këtu çmim të përshkallëzuar për një numër maksimal të përdorimeve. Kjo mënyrë licencimi është e përshtatshme për kompanitë më pak të njohura.
- Çmimi fillestar është shumë më i ulët për Bing Maps sesa për Google Maps.
- Garanton që nuk shfaqin reklama në hartë pa lejen tuaj paraprake.

(Marshall, 2010)

Pra, të dyja API-t kanë oferta shumë të ngjashme edhe pse Google nuk jep kufizim të qartë në lidhje me numrin maksimal të përdorimeve brenda vitit apo ditës. Pra, Google Maps API duhet të merret në konsideratë nëse ju dëshironi të përdorni versionin falas dhe atë mbi 125 mijë përdorime në vit. Nëse dëshironi të përdorni versionin me licencë, atëherë duhet t'i jepni përparësi Bing Maps ndaj Google Maps, meqë shërbimet e Bing Maps janë më fleksibile në përmbushjen e kërkesave në lidhje me projektet specifike.

Një kështillë e fundit, të jeni të kujdesshëm në përdorimin e licencës falas. Falas nuk është e mundshme të jetë gjithë jetën pa ndonjë ndryshim të mundshëm. Google ka një rrjetë reklamash në të gjithë botën dhe që punon në të gjitha platformat, mirëpo deri me tani nuk është munduar ta aplikojë. Bing Map nuk ka provuar kurrë ta bëjë një gjë të tillë, por kjo nuk do të thotë se në të ardhmen nuk do ta bëjë një gjë të tillë. Sido që të jetë, nëse jeni duke planifikuar një model biznesi për faqen tuaj, që mbështetet në hartat falas, dhe ky plan

është afatgjatë apo afatmesëm, duhet ta keni në mendje që përdorimi falas nuk do të jetë përgjithmonë. (Marshall, 2010)

Krahasimi mes Open Street Map dhe Google Maps

OpenStreetMap

OpenStreetMap u shfaq për herë të parë në vitin 2004 në tregun e Britanisë së Madhe dhe paraqet hartën botërore si shërbim falas. Ajo u krijua për të vetmen arsye: meqë përdoruesit britanikë ishin të pakënaqur me entin e matjes së vendit dhe filluan shërbimin e tyre në lidhje me hartat. Të dhënat e GPS-së grumbullohen nga vullnetarët dhe i ofrohen secilit falas. Shërbimi ofrohet në principin e Wikipedias, në mënyre që secili të mund të kontribuojë në korrigjimin e gabimeve eventuale. (TWT, 2015)

Një qasje e tillë u adaptua dhe u përshtat edhe nga Google, në mënyre që ta rrisë dhe ta përmirësojë përvojën e përdoruesve në shërbimet e tij në lidhje me hartat. Google-i po ashtu në vitin 2008 implementoi dhe shënjuesit (markers), dhe kështu Google mori pamje dhe qasje të ngjashme ashtu si OpenStreetMap. (TWT, 2015)

Komuniteti i madh i bën shërbime falas të OpenStreetMap-it unik. Mijëra njerëz, që punojnë në përditësimin e kësaj harte, do të pasurojnë jetën e përdoruesve në mbarë botën. Megjithatë, harta e hapur e botës sjell mangësi. Në veçanti, vendet më të vogla ose më të qeta shpesh nuk plotësohen mirë, për shkak të pamjaftueshmërisë së vullnetarëve që kontribuojnë për këto rajone. Kjo mangësi kontribuon në dobësimin e cilësisë së shërbimit të ofruar. (TWT, 2015)

OSM mund të përdoret edhe nga kompanitë në vend të skicave të udhëtimit dhe hartave të qyteteve të ndryshme. Qyteti i Hamburgut përdor këtë shërbim, për shembull, në përpilimin e një harte falas të qytetit dhe lejon përdoruesit të bashkëpunojnë dhe të rishpërndajnë hartën. Kjo natyrisht do të thotë se institucionet dhe kompanitë nuk kanë asnjë kontroll mbi hartat e tyre dhe se çfarë është ndryshuar nga ana e përdoruesit. (TWT, 2015)

Google Maps

Google Maps konsiderohet si ofruesi numër një në ofrimin e shërbimit të hartave. Prezantimi i qartë, rrugët për këmbësorët, për makina dhe për çiklistët dhe transportin publik e bëjnë shërbimin e Google-it kaq të dashur dhe të njohur. Avantazhi i qartë i Google Maps është që ofrimi i shërbimit është ideal për përdorim të korporatave të internetit dhe intranetit. Google Maps API është i përshtatur për nevojat e biznesit, dhe kështu lejon futjen e të dhënave personale apo të hartave të tjera si mbulesë, mbi hartat e vetë Google-it. Platforma e Google-it përmes imazheve satelitore lejon ndërtimin e aplikimeve atraktive, gjë e cila është përparësi shumë e madhe ndaj OpenStreetMap. Kjo platformë përfshin drejtimet, funksionet për analiza dhe një bazë të madhe të të dhënave me informacione në lidhje me lokacionet e ndryshme. (TWT, 2015)

OpenStreetMap është veçanërisht e përshtatshme për përdorim privat dhe është projektuar në mënyrë interaktive. Ndërsa Google Maps ofrojnë ndërfaqe më të mirë për përdoruesin si dhe është e përshtatshme për vizualizimin e të dhënave të kompanive.

(TWT, 2015)

Open Street Map	Google Maps
Qasje falas në hartën botërore.	E njohur dhe e lehtë në përdorim.
Hartat mund të ndahen edhe me të tjerët.	Cilësi e lartë imazhesh.
Funksionon si në ueb brouser ashtu dhe në pajisje mobile me sistem operativ (Android dhe IOS).	Prezantimi i qartë, rrugët për këmbësorët, makina dhe çiklistët dhe transportin publik.
Mund të jepet adresa e cakut dhe të planifikohet rruga.	Përparësia qëndron te pajisjet fundore mobile.
Përdorues nga e gjithë bota mund ta përpunojnë hartën duke futur dhe korrigjuar shënime.	Google Maps API është i përshtatshëm për përdorime biznesi.
Qasja Crowd-Sourcing.	Zhvillimi i aplikacioneve bazuar në vendndodhje.
	Përdorimi përmes brouser-it, pajisjeve mobile dhe me porosi.

Tabela 13: Open Street Map vs Google Maps

3.5. Kompjuterizimi në Re

Kompjuterizimi në Re është një hap i suksesshëm në zhvillimin e përgjithshëm të internetit dhe modeleve të ndryshme të biznesit, përkundër pikave të diskutueshme që ka. Numri i madh i shërbimeve të tilla, të ofruara në të gjithë globin është një tregues që vërteton suksesin e kësaj teknologjie. (Musafi)

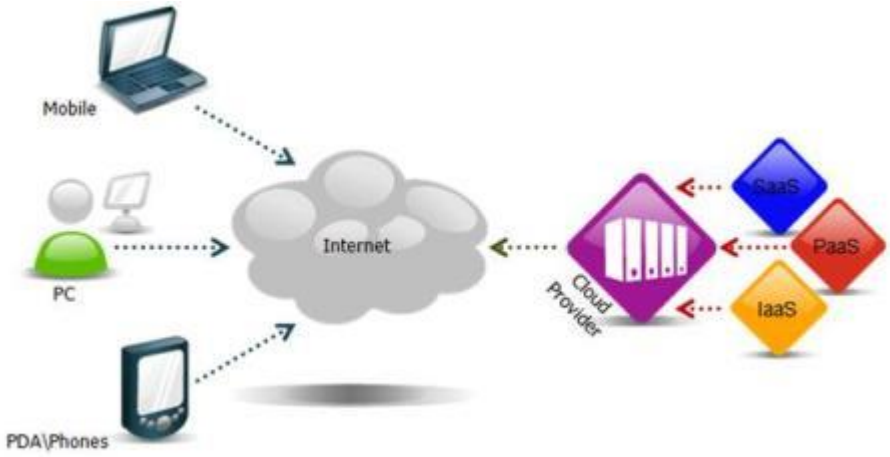


Figura 6: Kompjuterizimi në Re *(Musafi)*

Shërbimet cloud janë popullarizuar për shkak se ato mund të zvogëlojnë shpenzimet dhe kompleksitetin e mbajtjes së kompjuterëve dhe rrjetave. Meqenëse përdoruesit nuk kanë për të investuar në infrastrukturë të TI-ve, blerjen e pajisjeve, programeve apo edhe licencave, përfitimet janë se në fillim ka kosto të ulët, përdorim fleksibël, zgjidhje, shpërndarje e shpejtë dhe kthim të shpejtë në investime. (Musafi)

Sipas Institutit Kombëtar Amerikan të Standardeve dhe Teknologjisë (NIST) definicioni i kompjuterizimit në Re është formuluar kështu: "*Cloud computing është një model që bën të mundur përshtatjen, mbikërkesën e qasjes në rrjetë në një pishinë të përbashkët të burimeve të informacionit të konfigurueshme (p.sh., rrjetet, serverët, ruajtja, aplikimet dhe shërbimet) që mund të përgatiten më shpejt dhe të lëshohet me përpjekje minimale të menaxhimit ose ndërveprim nga ofruesi i shërbimit. Ky model promovon mundësitë dhe përbëhet nga pesë karakteristika thelbësore, tri modele të shërbimit dhe katër modele të shpërndarjes*." (Mell & Grance, 2011)

Kjo nënkupton shfrytëzimin si shërbim të burimeve të teknologjisë informative siç janë pajisjet teknologjike (harduer) dhe softuerët. Ky lloj shërbimi tanimë është i njohur për ne

sepse jemi shfrytëzues të një shërbimi të tillë si email-i, qasja në rrjetet sociale, ngarkimi i fotove apo shënimeve të tjera në shërbime si Onedrive apo Google Drive, etj. (Musafi)

Ndërkohë që Reja kompjuterike ka shumë premisa për të gjitha ndërmarrjet e reja që kërkojnë zgjidhje informatike, ajo në të njëjtën kohë ka edhe një numër mangësish të natyrshme të tilla si:

- Përshkallëzim i kufizuar për rastin e një Reje si ofrues i vetëm. Edhe pse shumica e ofruesve të ndryshëm të Reve kompjuterike pretendojnë se ka një përshkallëzim infinit, në realitet është më se e arsyeshme të supozohet se me rritjen e raportit të shfrytëzimit të resë, ofruesit më të mëdhenj mund të fillojnë të përballen me problemet e përshkallëzimit. Në një kohë më të gjatë, problemi i përshkallëzimit mund të pritet që të përkeqësohet, pasi ofruesit shërbejnë një numër gjithnjë e në rritje të shërbimeve online, ku secili prej tyre aksesohet nga një numër i madh përdoruesish globalë në çdo kohë. (Musafi)
- Mungesa e ndërveprimit ndërmjet ofruesve të Reve. Teknologjitë bashkëkohore të Reve, kur janë projektuar, nuk kanë pasur në mendje ndërveprimin midis tyre. Kjo gjë parandalon infrastrukturën e reve të ofruesve të vegjël e të mëdhenj që të hyjnë në tregun e sigurimit të Reve. Në përgjithësi, kjo e "mbyt" konkurrencën dhe i detyron konsumatorët drejt një shitësi të vetëm, pra drejt një monopoli. (Musafi)

Përparësitë e Kompjuterizimit në Re

Sipas (Musafi) përparësitë e Kompjuterizimit në Re janë:

- ***Efikasiteti*** **–** i kësaj teknologjie është shumë më i madh sesa i asaj lokale meqë kompanitë investojnë në pajisjet më të reja teknologjike në mënyrë që shërbimin ta bëjnë sa më efikas.
- ***Qasja e konsumatorëve*** **–** realizohet përmes veprimeve të thjeshta, të cilat janë të mbështetura nga teknologji të sofistikuara, që mundësojnë realizimin e veprimeve në mënyrë të përnjëhershme duke mos e ndjerë shfrytëzuesi se është duke vepruar me aplikacion në largësi.
- ***Mirëmbajtja*** **–** është shumë më e lehtë sepse aplikacionet e Kompjuterizimit në Re nuk duhen instaluar në mjetin e secilin shfrytëzues, por vetëm në teknologjinë e kompanisë, gjë e cila lehtëson dukshëm mirëmbajtjen dhe përditësimin e saj.
- ***Shpenzimet*** **–** janë më të ulëta si për shfrytëzuesit e shërbimeve po ashtu edhe për ofruesit e tyre në Re. Shfrytëzuesit e shërbimeve, përmes një investimi dukshëm më të ulët në marrjen e shërbimeve, lirohen nga barra e teknologjisë dhe nga zhvillimet e saj dhe përqendrohen te biznesi i tyre ose tek ideja bazë për të cilën u nevojitet kjo teknologji, kurse ofruesit e kësaj teknologjie kanë ofrimin e saj kanë bazë të biznesit për atë dhe specializohen në bashkëveprimin me të dhe në zhvillimin e saj, gjë që mundëson rritjen e vlerave të diturisë dhe njëkohësisht uljen e shpenzimeve nga specializimi. Për konsumatorët uljen e shërbimeve e mundëson dukshëm blerja e shërbimeve sipas kërkesës, ku ata paguajnë për atë që shfrytëzojnë dhe asgjë më tepër. Në anën tjetër, uljen e shpenzimeve, gjithashtu, e mundëson edhe shfrytëzimi i shumëfishtë që nënkupton shfrytëzimin e së njëjtës teknologji, nga ana e shumë konsumatorëve, të ofruar nga një kompani, ku të njëjta shpenzime ndahen në shumë pjesëmarrës.

- ***Pavarësia e vendit nga shërbimi*** **–** është një karakteristikë që e bën shumë tërheqës këtë teknologji meqë e bën shfrytëzuesin e saj të pavarur nga vendi dhe mjeti. Shfrytëzuesi mund të realizojë qasjen në shërbimet e ofruara nga cilido vend me cilindo mjet duke mos pasur asnjë pengesë si në qasje ashtu edhe në ushtrimin e veprimeve.
- ***Paraqitja virtuale*** **–** është veprim të cilin e realizojnë kompanitë ofruese të shërbimeve për të rritur efikasitetin e shërbimeve dhe në raste të veçanta edhe për të respektuar ligjet dhe rregulloret të cilat përcaktojnë kufijtë e të drejtave të të dhënave. Paraqitja virtuale nënkupton shpërndarjen e serverave dhe teknologjisë fizikisht në atë mënyrë që shërbimet dhe ngarkimi i serverave të realizohet në mënyrë optimale.
- ***Siguria*** – është një element i rëndësishëm për të dhënat e konsumatorëve, e posaçërisht të atyre komercialë. Ofruesit e kësaj lloj teknologjie janë të specializuar dhe shpesh herë ofrojnë zgjedhje shumë më të sigurt nga ajo që do të arrinin konsumatorët nëse do të investonin në një zgjedhje të veten. Kuptohet që siguria ka edhe perspektiva tjera, por që në këtë rast në të gjitha elementet jam munduar të sjellë vetëm një perspektivë në dukje.
- ***Karakteristika të tjera*** – sigurisht që ekzistojnë edhe karakteristika të tjera të cilat e veçojnë këtë teknologji e që ne nuk i kemi marrë parasysh këtu e për të cilat mund të mësoni nga burime të shumta në internet.

Modelet e shërbimit të Kompjuterizimit në Re

Modelet e shërbimit të cloud computing janë Softueri si shërbim (SaaS), Platforma si shërbim (PaaS) dhe Insfrastruktura si shërbim (IaaS). Softueri si shërbim (SaaS) – është i dizajnuar për klientët fundorë që ofrohet dhe shpërndahet përmes internetit.

Me SaaS, një ofrues i lejon klientit një aplikacion varësisht prej kërkesave tij, nëpërmjet një abonimi. Ky shërbim është në rritje të shpejtë dhe parashikohet të ketë rritje në vazhdim. Nga kjo shohim se shërbimi SaaS do të jetë i kërkueshëm dhe i domosdoshëm për çdo organizatë. Karakteristikat e softuerit si shërbim janë: mund të menaxhohet nga një vend, është shërbim që ofrohet si model një për të gjithë, përdoruesit nuk kanë nevojë të paguajnë licenca apo të bëjnë përditësime për aplikacione. (Reiter, 2016)

Platforma si shërbim (PaaS) – Në modelin PaaS, platformat kompjuterike ofrohen si shërbim për të shpërndarë dhe për të zhvilluar aplikacionet e përdoruesit. Ofrohet një mjedis programues dhe me kujdes për të përkrahur zhvillimin e aplikacioneve dhe shpërndarjen te përdoruesit e organizatës ose të kompanisë. Në dallim me modelin SaaS, që ofrohet si softuer përmes shfletuesve të internetit (ueb faqeve), PaaS është një platformë për krijimin e softuerëve dhe gjithashtu ofrohet përmes internetit dhe ueb faqeve. Karakteristikat e platformës si shërbim janë: Shërbehet për të zhvilluar, testuar, shpërndarë, menaxhim dhe mirëmbajtje të aplikacioneve në atë mjedis të zhvillimit. Paas përfshin disa zgjidhje si: planifikimin e projekteve dhe të mjeteve të ndryshme të komunikimit. I ndërtuar për t'u zgjeruar pa problem dhe integrim i lehtë me shërbime dhe baza të dhënave të tjera. (Reiter, 2016)

Infrastruktura si shërbim (IaaS) – Është një mënyrë e furnizimit me infrastrukturë dhe pajisje të TI-së (p.sh. server, hard disk hapësirë, sisteme operative) që janë dhënë si shërbim sipas dëshirës për kompanitë përmes mjedisit virtual cloud. Klientët mund t'i shfrytëzojnë këto burime në vend që të blejnë servera, programe, qendra të mëdha për të dhënat, hapësira dhe pajisje të shumta të rrjetave. Iaas mund të përdoret si infrastrukturë publike ose private apo edhe të dyja bashkë. Kjo qasje e kombinimit quhet hibrid cloud. Karakteristikat e infrastrukturës si shërbim janë: Në krahasim me dy modelet tjera Saas dhe Paas, Iaas është një model që zhvillohet shpejt. Të thënat e lartpërmendura janë disa nga karakteristikat e Iaas që është pranuar në përgjithësi duke u përshtatur, që resurset i shpërndan si shërbim, ka mundësi të mëdha zgjerimi dhe shkallëzimi dinamik, me çmime të ndryshme varësisht nga modeli që kërkohet dhe që mundëson përfshirjen e shumë përdoruesve në një harduer. (Reiter, 2016)

Modelet e shpërndarjes së shërbimeve të Kompjuterizimit në Re

Modelet e shpërndarjes cloud, sigurojnë mënyrën sesi këto shërbime mund të shpërndahen në mes të organizatave. Klientët, që kërkojnë zgjidhje cloud, duhet të zgjedhin modelin e shpërndarjes që i përshtatet për biznesin e tyre, kërkesave dhe kushteve teknike. Dallojmë katër modele primare të shpërndarjes: private cloud, public cloud, community cloud dhe hybrid cloud. (Hurwitz, Bloor, Kaufman, & Halper, 2009)

Private Cloud – është infrastruktura në pronësi të organizatës ose të kompanisë dhe menaxhohet nga një palë e tretë. Zakonisht ky model i shpërndarjes i shërben një numri të caktuar shfrytëzuesish, të cilëve u garanton menaxhim dhe kontroll varësisht nga kërkesat e

tyre. Private cloud ndahet bashkë në mes të departamenteve, punëtorëve dhe lokacioneve të kompanisë ose organizatës përmes një mënyre të limituar. Ky model është ideal për veprime që kërkojnë siguri të lartë. (Hurwitz, Bloor, Kaufman, & Halper, 2009)

Public Cloud – Shërbimet e ofruara për këtë model janë përmes internetit dhe udhëhiqen nga kompanitë furnizuese cloud (provider). Publik cloud është i mundur për të gjithë shfrytëzuesit publikë me apo pa pagesë. Shumë organizata janë të gatshme për përqafuar këtë model, sepse ofron përfitime jo vetëm për sa i përket fleksibilitetit dhe zgjerimit, por edhe shpejtësisë dhe kostos së lirë. Megjithatë disa nuk preferojnë t'i lëvizin disa lloje të të dhënave në public cloud për shkak të shqetësimeve që kanë të bëjnë me sigurinë. Disa nga kompanitë që e ofrojnë këtë model janë Microsoft, Amazon, Google etj. (Hurwitz, Bloor, Kaufman, & Halper, 2009)

Hybrid Cloud – është kombinimi i dy apo më shumë modeleve si për shembull private dhe publik cloud. Edhe pse ka një përbërje prej dy ose më shumë modeleve, mbeten subjekte të veçanta, por të lidhura së bashku nga pronari ose nga standardet dhe mundëson transport të lehtë të të dhënave dhe aplikacioneve. (Hurwitz, Bloor, Kaufman, & Halper, 2009)

Pavarësisht rreziqeve të shumta nga public cloud, cloud computing konsiderohet ende një trend premtues. Organizatat ose kompanitë duhet të mbajnë kontrollin mbi shënimet e rëndësishme, shërbimet apo infrastrukturën. Për këtë kemi kombinimin e modeleve të shpërndarjes private dhe publike që njihet si cloud hibrid. Të dhënat e rëndësishme apo shërbimet kritike mund të përdoren përmes rrjetit të brendshëm intranet, që na jep siguri dhe rrjet të qëndrueshëm. Ndërsa shërbimet jokritike apo të dhënat e tjera mund të vendosen tek ofruesit e jashtëm, që sigurisht janë të ekspozuar nga rreziqet e ndryshme,

sepse përdorimi aty bëhet ngado edhe në mënyrë mobile dhe ka nivel të ulët të sigurisë. Hybrid cloud kujdeset që të balancojë shpenzimet afatshkurtra dhe ato afatgjata. (Baun, Kunze, Nimis, & Tai, 2011)

Karakteristikat e hybrid cloud

Kombinimi unik i arkitekturës private dhe publike na prezanton disa çështje specifike, duke trashëguar karakteristika të ndryshme nga të dyja modelet si: lidhja mes tyre, dobitë e ndryshme të saj, zgjerim i lehtë, lehtë për të punuar etj. (Baun, Kunze, Nimis, & Tai, 2011)

Dobitë e hybrid cloud

Sistemet e bazuara në cloud hibrid kanë shumë dobi dhe përparësi. Pjesa më e madhe trashëgohet nga sistemi privat, ku dihet që është shumë i sigurt dhe kompanitë ose organizatat kontrollojnë dhe menaxhojnë shërbimet e tyre të vlefshme. Derisa sistemi privat është më i sigurt, ai publik i ekspozon kompanitë ndaj rreziqeve të mëdha të sigurisë. Përfitimet e sistemeve hibride cloud lidhen me veçoritë e tyre strukturore si: zgjerimi, ngjashmëritë, ulje të ngarkesës dhe me aspektet si shpejtësia dhe përditësimet. (Baun, Kunze, Nimis, & Tai, 2011)

Shpenzimet e Hybrid Cloud

Në sistemet hibride cloud shpenzimet janë të ndara në mes të pjesëve private dhe asaj publike. Sistemet private cloud janë të kushtueshme në fillim sepse duhet të investohet në infrastrukturë, ndërsa më pas ato kanë shpenzime efektive në planin afatgjatë. E kundërta është me sistemet publike sepse ato kanë shpenzime të mëdha në planin afatgjatë, por nuk

kërkohet investim i madh në fillim. Siç e cekëm edhe më herët, sistemet e përbashkëta hibride cloud mundësojnë fleksibilitet më të madh në balancimin e shpenzimeve. Organizatat ose kompanitë mund t'i menaxhojnë vetë shpenzimet dhe të zgjedhin mënyrën e pagesës për sistemet publike sipas kërkesës apo shërbimeve që dëshirojnë të përdorin. (Baun, Kunze, Nimis, & Tai, 2011)

Objektivi strategjik për menaxherët e teknologjisë së informacionit është që të arrijë efikasitetin e plotë të sistemeve private cloud brenda pak viteve. Prandaj çdo ditë e më tepër shumë kompani dhe organizata po preferojnë aftësinë e sistemeve hibride cloud. Në bazë të shfrytëzimit, pra të shpenzimeve të bazës së të dhënave dhe shpenzimeve të ofruesve të cloud, bizneset duhet të zgjedhin nëse është e leverdishme apo jo kalimi në cloud. Shpenzimet në sistemet publike janë nga më të ndryshmet, por zakonisht ato përfshihen të gjitha në çmim nga ofruesi, si për shembull: shfrytëzimi i qendrave të të dhënave, shpenzimet operative, mirëmbajtja, rryma, ftohja e sistemeve të mëdha, dislokimi i tyre në rast nevoje etj. Varësisht nga kapaciteti i qendrës së të dhënave lokale dhe prej shpenzimeve të ofruesve të jashtë cloud, bizneset duhet të vendosin sa burime të jashtme të përdorin. Sot, shpenzimet operacionale harduerike kushtojnë lirë dhe ndërrimi i pajisjeve të dalë jashtë funksionit apo elementet e tjera në qendrën e të dhënave mund të bëhet edhe nga një staf i trajnuar pak. (Baun, Kunze, Nimis, & Tai, 2011)

Lidhja midis SOA-s dhe Kompjuterizimit në Re

SOA dhe cloud computing janë aktivitete komplementare dhe të dyja do të luajnë role të rëndësishme në planifikimin IT për ekipet e udhëheqjes së lartë për vitet që do të vijnë.

Cloud computing nuk e zëvendëson SOA, ose përdorimin e komponentëve të softuerit të shpërndarë, si një teknologji integruese. Nevoja për të përkrahur integrim më të gjerë dhe më konsistent të sistemeve do të vazhdojë. Prirja nga ekipi i lidershipit për të marrë në konsideratë aftësitë IT si një mall do të vazhdojë të ulë presionin për buxhetin IT. Cloud computing dhe SOA nuk janë sinonime, ndonëse ato ndajnë shumë karakteristika. Zgjedhja e njërit nuk e plotëson tjetrin. P. sh.: integrimi i vazhdueshëm i sistemit, ku softueri si komponent i shpërndarë ose shërbimet (SOA) në thelb nuk e virtualizon harduer-in, ose nuk e transferon shtresën e prezantimit te një palë të tretë ofruese (cloud computing). Përmbushja e transferimit të suksesshëm të funksioneve IT të artikullit (cloud computing) nuk integron sistemet me porosi në biznes, ose të agregojë të dhëna në një shfaqje të vetme "mash-up" (SOA). (Fejzaj, 2014)

Kompjuterizimi në Re dhe virtualizimi

Në thelb virtualizimi ndryshon nga cloud computing sepse virtualizimi është softuer që manipulon harduer-in, ndërsa cloud computing i referohet një shërbimi i cili është rezultat i manipulimit. (Fejzaj, 2014)

Cloud Computing është përtej teknologjisë së virtualizimit. Virtualizimi është i lidhur me kushtet në optimizimin e burimeve të infrastrukturës IT. Virtualizimi është një teknologji e

përdorur në konceptin e Cloud Computing. Virtualizimi është duke përdorur të njëjtën infrastrukturë harduer për të ndërtuar disa servera për kërkesat. Virtualizimi përdor të njëjtën infrastrukturë harduer për të ndërtuar disa servera virtualë për të cilët ka nevojë. Në qoftë se kjo vendoset në një arkitekturë me shtresa, shtresa 1 do të jetë SAN (Storage Area Network), shtresa 2 do të jetë harduer servers (blade servers) për shpërndarjen e burimeve dhe shtresa top do të jetë host server. Softuerë-t e virtualizimit si Citrix, VMware's vSphere, Xen, Microsoft Hyper V, Sun xVM do të ekzekutohen në serverat e shtresës top të cilët quhen host servers. (Kalavace, 2013)

Teknikat që kontribuojnë në teknologjinë e reve kompjuterike

Ardi Benusi në disertacionin e tij (Benusi, 2015) numëron teknologjitë e mëposhtme si kontribuuese në teknologjinë e reve kompjuterike.

Virtualizimi

Termi i referohet një mjedisi në gjendje që t'u sigurojë përdoruesve të gjitha shërbimet e nevojshme që mbulohen nga harduer.

Ueb shërbimet dhe SOA

Ueb shërbimet ofrojnë shërbime në internet duke përdorur teknologji si XML, Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) etj. Organizimi i këtyre shërbimeve menaxhohet në formën e një arkitekture të orientuar

shërbimesh (SOA) nëpërmjet së cilës një detyrë e caktuar kryhet duke përdorur shumë shërbime.

Web 2.0

Web 2.0, është konsideruar si një teknologji që iu lejon përdoruesve të krijojnë faqe interneti, që nuk kufizohen vetëm në leximin e tyre; në fakt u mundëson përdoruesve një ndryshim dinamik duke u krijuar atyre një platformë më krijuese.

Disa nga ofruesit e shërbimeve në retë kompjuterike janë: Amazon EC2, Amazon S3, SQS, Google, Microsoft, Salesforce.com, Sun Microsystems etj dhe secili ka karakteristikat sipas klasifikimeve të bazuara në strukturën e shërbimeve: private, publike dhe hibride.

Individë dhe organizata që përdorin internetin gjithmonë e më shumë kërkojnë të jenë anonim. Është e rëndësishme që blerjet online, dërgimi i email-eve të mos ua ekspozojnë të tjerëve identitetin, interesat dhe aktivitetet e tyre. Gjithashtu korporatat dhe organizatat ushtarake duhet të komunikojnë me organizata të tjera pa ua shfaqur ekzistencën e këtyre komunikimeve konkurrentëve ose "armiqve".

Skemë shifrimi

Një skemë shifrimi për fshehtësinë e të dhënave. Në retë kompjuterike duhen të dallohen çështjet e privatësisë së të dhënave në transit dhe atyre të ruajtura në disqet e ngurta.

Ofruesve të shërbimeve në retë kompjuterike duhet t'u jepet qasje e kufizuar në të dhënat e përdoruesve; vetëm menaxhim i tyre pa pasur mundësi të shohin dhe të lexojnë përmbajtjen e informacionit.

Backup i të dhënave

Backup i të dhënave dhe replikimi i tyre duke siguruar rikuperim të shpejtë të informacionit të humbur në rast fatkeqësie.

Me rritjen e dukshme të migrimit të infrastrukturës në një re kompjuterike, çështjet e sigurisë dhe privatësisë janë bërë më të sofistikuara. Nëpërmjet rritjes së kërkesave për qasje në aplikacione të ndryshme, mundësia për sulme kibernetike gjithashtu rritet. Individë të ndryshëm, në mënyrë të shpeshtë duhet të japin online kredencialet për identifikimin e tyre dhe gjatë këtij procesi, një ndërhyrës i paautorizuar mund të "vjedhë" informacione të rëndësishme të përdoruesit. Skemat e mëposhtme duhet të jenë të ndërtuara në mënyrë që të mbulohen çështje të tilla të sigurisë dhe privatësisë si: konfidencialiteti, integriteti, autentikimi, rikuperimi i të dhënave në rast fatkeqësie. (Tianfield, 2012)

Virtualizimi

Kompjuterët e para pak viteve, ishin të tillë që harduer-i i tyre ishte dizenjuar në mënyrë të atillë, që të punonte vetëm në një sistem operimi, dhe në një program të vetëm. Duke bërë kështu, që shumë makina të mos përdoreshin më, pra të liheshin në harresë. Pikërisht, virtualizimi bëri që këto "makina" të ktheheshin përsëri në jetë. Virtualizimi na lejon që të përshtasim një numër të konsiderueshëm të makinave virtuale në një makinë të vetme fizike (me termin 'makinë', i referohem kompjuterit si tërësi harduer-i). Një mënyrë e tillë, pra duke përdorur virtualizimin, secila makinë virtuale, e instaluar tashmë në kompjuterin tonë, do të ndajë të gjithë burimin e saj me makinat e tjera. Vlen,

jashtëzakonisht shumë, fakti që, pavarësisht ndarjes së burimeve, kapaciteti i pajisjeve harduer të kompjuterit nuk mund të ndryshohen, si p. sh. kapaciteti i CPU-së, RAM-i etj. (Fejzaj, 2014)

Përpos kompjuterit dhe lidhjes së internetit, virtualizimi është teknologjia më e rëndësishme e cila e lejon kompjuterizimit në Re për të arritur potencialin e saj të plotë (Böhm, Leimeister, Riedl, & Krcmar, 2014). Duke falënderuar virtualizimin, një server fizik mund të ndahet ne disa servera virtualë, në këtë mënyrë del një shërbim krejtësisht i ri. Këto servera virtualë quhen "raste virtuale" (Zhang, Zhang, Chen, & Wu, 2010)

Praktikisht, virtualizimi siguron aftësinë për të ekzekutuar aplikacione, sisteme operative, ose sisteme shërbimesh në një mjedis të veçantë logjik të pavarur nga sistemi fizik i kompjuterit. (Pëllumbi, 2014)

Virtualizimi është simulimi i një platforme harduerike, një sistemi operativ, një storage device apo simulimi i burimeve të rrjetit. Me fjalë të tjera është krijimi i një versioni virtual të tyre.

Platforma e virtualizuar performohet në një platformë harduerike të dhënë nga softueri host (një program kontrolli), që krijon një mjedis kompjuterik artificial, një makinë virtuale (VM), për softuerin guest të tij. (Pëllumbi, 2014)

Fokusimi i virtualizimit në mjediset operative logjike në vend të atyre fizike i bën aplikacionet, shërbimet dhe instancat e një sistemi operativ të transferueshëm nëpër sisteme fizike të ndryshme kompjuterësh. Platforma e virtualizimit performohet në një platformë harduer të caktuar nga *hosti softuer* (ose programi i kontrollit), e cila krijon një kompjuter të simuluar, një makinë virtuale për *guest softuerin*. (Pëllumbi, 2014)

Memoria virtuale është një teknikë menaxhimi e memories, e cila bën që një sistem operativ të përdorë segmente të ndara të memories si një hapësirë memorie të vetme e të vazhdueshme. Memoria virtuale është implementuar tradicionalisht në një sistem operativ nëpërmjet "paging", i cili bën që sistemi operativ të përdorë një file ose pjesë të dedikuara të disa pajisjeve të ruajtjes së të dhënave për të ruajtur faqe të memories të cilat nuk janë përdorur së fundmi. E njohur edhe si "pagingfile" ose "swap space", sistemi mund të transferojë shumë shpejt faqe drejt dhe prej kësaj zone duke qenë se sistemi operativ ose një aplikim ekzekutues kërkojnë t'i qasen përmbajtjes së këtyre faqeve. Sistemet operative si UNIX (duke përfshirë Linux, sistemetoperative BSD, dhe Mac OSX) edhe Microsoft Windows përdorin disa forma të memories virtuale për të bërë të mundur që sistemi operativ dhe aplikacionet t'i qasen më shumë të dhënave sesa mund të vendosen në memorien fizike. (Pëllumbi, 2014)

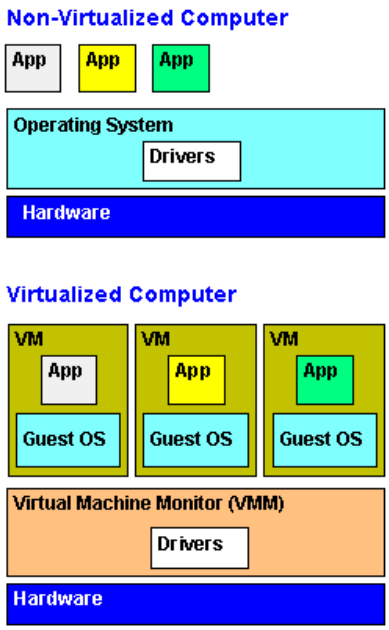


Figura 7: Arkitektura e Virtualizimit (Kalavace, 2013)

Lista e mëposhtme tregon informacionin, që duhet mbledhur për secilin nga aplikacionet,

që do të kalohet në makinë virtuale. Kjo jo vetëm që do të japë një referencë të mirë për informacion përkrahjeje dhe licencimi, por mund të ndihmojë edhe të identifikohen aplikacionet që janë kandidatë për t'u kaluar në makina virtuale. (Kalavace, 2013)

Aplikacioni dhe versioni – Emri dhe numri duhet të jetë specifik i versionit të aplikacionit.

Sistemi operativ aktual dhe versioni – Sistemi operativ në të cilin aplikacioni ekzekutohet.

Korrigjimet apo paketat e shërbimit të sistemit operativ –Çdo korrigjim specifik, që është aplikuar te sistemi operativ, përfshirë dhe paketat e shërbimit për aplikacionet e bazuara në Windows.

Sisteme operative të tjera të pranueshme –Sisteme të tjera operative dhe numrat shoqërues të versioneve që ky version i këtij aplikacioni mund të përdorë. (Kalavace, 2013)

Përparësitë e virtualizimit në Kompjuterizimin në Re

Për një ofrues të shërbimeve TI-së, përdorimi i virtualizimit ka një numër të madh të përfitimeve dhe avantazheve. Autorët (Baun, Kunze, Nimis, & Tai, 2011) i ndajnë dhe i përshkruajnë ato si më poshtë:

- Përdorimi i burimeve: serverat fizikë rrallë punojnë me kapacitet sepse operatorët e tyre në përgjithësi lejojnë burime të mjaftueshme për të mbuluar përdorimin e pikut. Nëse përdoren makinat virtuale, çdo kërkesë e ngarkesës mund të jetë e kënaqur me burimet e përbashkëta.

- Menaxhimi: Është e mundur për të automatizuar menaxhimin e burimeve të përbashkëta. Makinat virtuale mund të krijohen dhe konfigurohen automatikisht sipas kërkesës.

-

Konsolidimi: Klasa të ndryshme të aplikacioneve mund të konsolidohen për të lëvizur në një numër më të vogël të komponentëve fizikë. Përveç konsolidimit të serverit dhe magazinimit, gjithashtu është e mundur të përfshihen të gjitha pjesët e sistemit, të dhënat dhe baza e të dhënave, rrjeti dhe desktopi. Redukton koston dhe rrit efikasitetin.

- Konsumi i energjisë: Furnizimi me energji elektrike i qendrave të mëdha të të dhënave është bërë gjithnjë e më i vështirë dhe duke e parë gjatë periudhave më të hershme, kostoja e energjisë elektrike për të operuar një server është më e lartë sesa çmimi i blerjes. Konsolidimi zvogëlon numrin e komponentëve fizikë. Nga ana tjetër zvogëlon shpenzimet për furnizim me energji.

- Më pak hapësirë e nevojshme: Çdo metër katror i hapësirës së qendrës së të dhënave është e vogël dhe e shtrenjtë. Me konsolidimin, mund të kemi interpretim të njëjtë dhe të shmangim zgjerimin e kushtueshëm të një qendre ekzistuese.

-Plani i emergjencës: Është e mundur që të lëvizin makinat virtuale nga njëri grup i burimeve në tjetrin.

Mangësitë e virtualizimit

Autorja Inda Pëllumbi (Pëllumbi, 2014) i ndan dhe i përshkruan ato si më poshtë:

-Rrezikshmëri e lartë në një defekt fizik: Në virtualizim kemi një pikë të vetme dështimi ose defekti. Imagjinoni të keni 5 servera dhe kuptoni fare mirë se çfarë efekti do të kishte mbi këto 5 servera një defekt i vetëm fizik, duke bërë që të 5 serverat të dilnin offline.

- A është e njëjta performancë? Për këtë ka një sërë dyshimesh, por e konkretizojmë si shembull. Le të marrim një RAM 4 GB dhe 2 CPU virtuale me nga 3 GHz, në një server virtual, i cili do të përdoret si një ueb server në zyrën tuaj. Pyetja është: a do të mundej serveri dhe platforma e aplikacioneve të përshtatej në një server fizik me 4 GB RAM dhe 2 CPU me nga 3 GHz? Unë dyshoj për këtë.

- Nuk është teknikë e lehtë

- Nuk përkrahet nga të gjitha aplikacionet. Disa aplikacione të database-ve nuk janë akoma të përshtatshme për një virtualizim.

Tri format ekzistuese të virtualizimit të kategorizuara si: virtualizimi i makinës, virtualizimi i pajisjeve të ruajtjes së informacionit dhe virtualizimi i rrjetit, kanë çuar në lindjen e reve kompjuterike. P. sh.: një numër makinash fizike, me kapacitete mjaft të mira, por shumë pak të pashfrytëzueshme nga aplikimet e përdoruesve, mund të konsolidohen në disa makina me një shfrytëzim më efektiv.

Llojet e virtualizimit

Virtualizimi i sistemeve operative

Virtualizimi në nivel sistemi, shpesh i referuar si virtualizimi i sistemit operativ, përshkruan implementime të ndryshme të ekzekutimit të mjediseve të shumëfishta në një mjedis të vetëm të kernelit të sistemit operativ. Virtualizimi në nivel sistemi bazohet te koncepti i ndryshimit të rrënjës (chroot) i cili është i vlefshëm në të gjitha sistemet moderne të ngjashme me UNIX.

Gjatë procesit të butimit të sistemit, kerneli mund të përdorë filesystem -e rrënjë për

të ngarkuar driver -at dhe të performojë inicializime të sistemit në faza të hershme. Kerneli mund të udhëtojë në një filesystem tjetër rrënjë duke përdorur komandën chroot në mënyrë që të montojë një filesystem në disk si filesystemin rrënjë përfundimtar. Mekanizmi chroot, i përdorur nga virtualizimi në nivel sistemi, i mundëson sistemit të nisë serverat virtualë me proceset e tyre që ekzekutohen lidhur me direktoritë rrënjë të filesystemit të tyre. Veprimi brenda kufijve të direktorive të tyre rrënjë dhe filesysteme-ve parandalon që serverat virtualë t'i qasen fajllave në filesystemet e njëri -tjetrit, dhe rrjedhimisht siguron mbrojtje bazë nga shfrytëzimi i proceseve të ndryshme të serverit ose vetë serverit virtual. Edhe nëse një server në chroot kompromethohet ai ka qasje vetëm te fajllat që janë lokalizuar brenda filesystem -it rrënjë të tij.

Ndryshimi bazë midis virtualizimit në nivel sistemi dhe virtualizimit të serverit është se nëse mund të ekzekutohen sisteme operative të ndryshme në sisteme virtuale të ndryshme.

Virtualizimi i kapaciteteve të ruajtjes

Virtualizimi i kapaciteteve të ruajtjes është një abstraksion logjik për ruajtjen fizike. Ai është çelësi i krijimit të sasive fleksibël dhe të zgjerueshme të kapaciteteve ruajtëse për sistemet e sotme kompjuterike.

Virtualizimi i kapaciteteve ruajtëse ka qenë i përhapur prej shumë vitesh dhe mund të jetë i njohur për çdo njeri që ka punuar me RAID, volum e logjike në sisteme të tilla si: Linux apo AIX, ose me sisteme skedarësh të rrjetave si AFS dhe GFS. Te gjitha këto teknologji kombinojnë pajisjet e disqeve fizike të disponueshme, në grupe prej kapacitetesh të ruajtjes, që mund të ndahen në seksione logjike të njohur si volume në te cilin mund të

krijohet dhe të montohet çdo sistem skedarësh për përdorim në një sistem kompjuterik. Një volum është një ekuivalent logjik i një particioni disku. (Kalavace, 2013)

Elementet bazë që e bëjnë virtualizimin e kapaciteteve të ruajtjes kaq të pëlqyer në mjediset e kompanive në ditët e sotme është sepse mundësojnë një kapacitet efektiv të pafundmë qe limitohet vetëm nga numri dhe përmasa e pajisjeve të suportuara fizikisht nga sistemi host ose sistemi host i ruajtjes. Virtualizimi i kapaciteteve ruajtëse mundëson një sasi më të madhe të kapacitetit fizik të disponueshëm për sistemet individuale dhe u jep mundësinë sistemeve ekzistuese të zmadhohen për ta mbajtur atë informacion. Teknologji të tilla si *RAID* (Redundand Array of Inexpensive Disks) dhe volumet logjike, të siguruara nga Linux LVM, LVM2, si dhe paketat EVMS janë përgjithësisht të limituara për t'u përdorur në një sistem në të cilin pajisjet aktuale të ruajtjes janë të lidhura fizikisht. Disa kontrollues RAID janë me porta duale duke u lejuar shumë kompjuterëve qasje tek të njëjtat volume dhe sisteme skedarësh nëpërmjet kontrollerit RAID. (Kalavace, 2013)

Për të përdorur menaxhuesit e volumit logjik, duhen përcaktuar particionet e diskut që do të përdoren për vëllimet logjike, krijimi i vëllimeve logjike në atë pajisje fizike, dhe më pas krijimi i një sistemi skedarësh në vëllimet logjike. Më pas mund të montohen dhe të përdoren këto sisteme skedarësh njësoj sikurse të montoheshin dhe përdoreshin sisteme skedarësh që ishin krijuar në particione të disqeve fizike. (Kalavace, 2013)

Virtualizimi i serverit dhe i makinave

Virtualizimi i desktopit

Virtualizimi i desktop-it është koncepti i ndarjes së desktopëve logjikë nga makina fizike. Njëra formë e virtualizimit të desktopit, Infrastruktura Virtuale e Desktopit (VDI), mund të merret e mirë duke qenë si forma më e avancuar e virtualizimit harduer. Qëndron në ndërveprimin direkt të një kompjuteri 'host', me anë të tastierës, mausit dhe monitorit të lidhura me të. Përdoruesi ndërvepron me një kompjuter 'host', me anë të lidhjes së rrjetit (LAN ose Internet), duke përdorur një desktop të një kompjuteri tjetër ose një pajisje mobile. Kompjuteri 'host' në këtë skenar shndërrohet në një kompjuter server, i aftë për të mbajtur disa makina virtuale 'host', në të njëjtën kohë me disa përdorues. Për përdoruesit, kjo do të thotë që ato mund të kenë qasje në desktopin e tyre nga një vendndodhje tjetër. Përdorimi i Virtualizimit Desktop, i lejon kompanisë që të qëndrojë më shumë fleksibile, në çdo ndryshim të fundit të teknologjisë. Të paturit një desktop virtual, u lejon shumë zhvilluesve të implementohen shpejt dhe të ndihen ekspertë në atë që bëjnë. (Microsoft Virtual Server)

Virtualizimi i aplikacioneve

Virtualizimi i aplikacioneve është një term 'çadër', në teknologjinë softuer, që siguron portabilitet, menaxhim dhe kompatibilitet të aplikacioneve, duke i enkalupsuar ato në formën e nënvizuar të sistemeve operative, në të cilën ato janë ekzekutuar. Një virtualizim i plotë i aplikacioneve nuk është i instaluar në kuptimin e parë të fjalës, por ai ekzekutohet sikur të ishte i tillë. (Microsoft Virtual Server)

Përqasjet me të përhapura të virtualizimit të serverave dhe makinave sot janë:

Makina virtuale paralele: Disa sisteme fizike ose virtuale organizohen në një makinë virtuale të vetme duke përdorur softuer clustering si një Makine Virtuale Paralele (PVM). Cluster-i rezultues është i aftë të ekzekutojë CPU komplekse dhe llogaritje me të dhëna intensive në mënyrë bashkëpunuese. (Tafa, 2013)

Paravirtualizimi: Është metodë e përdorur nga XEN, Denali dhe Vmware. Është i bazuar në hypervisor, por pajisjet nuk emulohen. Gjithashtu procesorët duhet të kenë karakteristika të njëjta. Në këtë mënyrë, komunikimi midis aplikacioneve në makinat virtuale dhe pajisjet I/O realizohet përmes driver-ave virtualë I/O që ngrihen në pjesën e sipërme të hypervisor-it. Kjo gjë rrit shpejtësinë e komunikimit midis aplikacioneve dhe pajisjeve I/O. (Tafa, 2013)

GuestOS

ekzekutohen në DomainU në paralel me HostOS që po ekzekutohen në Domain0. Doamin0 është më i privilegjuari, që do të thotë se sistemet operative që ekzekutohen në të mund të administrojnë sistemet e tjera operative. Paravirtualizimi mund të sigurojë përmirësime të performancës krahasuar me përafrime të tjera të virtualizimit të serverit dhe të makinës sepse modifikimet e sistemit operativ bëjnë të mundur që ai të komunikojë direkt me hypervisor-in, dhe kështu nuk pëson ndonjë nga mbingarkimet e shoqëruara me emulacionin e kërkuar për makinat e tjera të bazuara në hypervisor. (Tafa, 2013)

Virtualizimi i plotë: Nga makinat virtuale janë shumë fleksibël. Është shumë i ngjashëm me paravirtualizimin. Virtualizimi i plotë gjithashtu përdor një hypervisor, por ndërfut kod në hypervisor, i cili emulon harduer-in e poshtëm kur është e nevojshme, duke u dhënë mundësinë sistemeve operative të pamodifikuara për t'u ekzekutuar në krye të hypervisor-it. Emulatori softuer krijon një shtresë, që zbut ndryshimet në pjesën harduer të

arkitekturave, dhe bën të mundur ekzekutimin e së njëjtës makinë virtuale në hoste të ndryshme, me arkitektura të ndryshme, pa krijuar probleme. Kjo gjë i jep fleksibilitetin e lëvizjes së gjithë makinave virtuale nga hosti në host në mënyrë shumë të lehtë, por rritet kosto e performancës për shkak të overhead-it nga shtresa emulatore. Virtualizimi i plotë është një model i përdorur nga serveri VMWare ESX. (Tafa, 2013)

Virtualizimi i rrjetit

Termi "virtualizim i rrjetit" përshkruan aftësinë për t'iu referuar logjikisht burimeve të rrjetit sesa t'i referohemi pajisjeve fizike specifike të rrjetit, konfigurimeve, ose grupit të makinave të lidhura. Ka filluar me nivele të ndryshme të virtualizimit të rrjetit, duke filluar nga një makinë e vetme, virtualizim i pajisjeve të rrjetit i cili bën që shumë makina virtuale të ndajnë një burim fizik të vetëm rrjeti, deri në konceptet *enterprise - level* si për shembull rrjete virtuale private dhe teknikat *edge-routing* dhe *enterprise-cor*e për krijimin e nënrrjeteve dhe segmentimin e rrjetave ekzistuese.

Xen mbështetet në virtualizimin e rrjetit përmes paketës *bridge-utils* për të bërë të mundur që makinat virtuale të duken sikur kanë adresa fizike unike (Media Access Control, ose adresa MAC) dhe adresat unike IP. (Tafa, 2013)

Barrierat në retë kompjuterike

Shumë kompani dhe organizata nuk e marrin dot vendimin për të kaluar shërbimet dhe të dhënat e tyre në retë kompjuterike. Kjo lidhet me mosbesimin që ata kanë për sa u përket

çështjeve të sigurisë dhe privatësisë. Disa nga kufizimet që justifikojnë mungesën e besimit janë:

Konfidencialiteti dhe siguria

Ofruesit e shërbimeve në retë kompjuterike këmbëngulin se makinat e tyre dhe informacioni që ruhet në to janë të mbrojtur mjaft mirë ndaj sulmeve dhe vjedhjeve të ndryshme. Gjithashtu ky informacion në Re është më i sigurt se në kompjuterët personalë të përdoruesve të ndryshëm.

Megjithatë ka pasur mjaft raste kur siguria është cenuar dhe i gjithë sistemi është bërë jofunksional për orë të tëra.

Vëzhgime mbi çështjet e sigurisë në retë kompjuterike

Një Re publike shërben si host për një numër të ndryshëm makinash virtuale dhe siguria në këtë tip reje është më e cenueshme, sepse me rritjen e përdoruesve, numri dhe natyra e rreziqeve janë më të mëdha. Migrimi në një Re private është një zgjedhje më e sigurt me mundësinë e një kalimi të mundshëm në një Re publike.

Rreziqet në retë kompjuterike

Përdorimi i SaaS në retë kompjuterike u bë i mundur nga zhvillimi i teknologjisë Web 2.0 duke i lehtësuar përdoruesit nga detyra të tilla si instalimi i programeve dhe mirëmbajtja e tyre. Në këto mjedise, siguria është bërë gjithmonë e më shumë një kërkesë parësore. Më

poshtë jepen disa nga elementet bazë të sigurisë, të lindura nga sulmet e ndryshme në teknologjinë Web 2.0. (Benusi, 2015)

Sulmet SQL injections, realizohen kur një kod dëmtues ndërfutet në një kod standard SQL dhe në këtë mënyrë një ndërhyrës i paautorizuar ndërhyn në databazë dhe akseson informacion sensitiv. (Benusi, 2015)

Sulmet me Cros Site Scripting (XSS), realizohen kur një skript dëmtues futet brenda përmbajtjes së faqes së internetit. Më të prekura janë faqet me përmbajtje dinamike të informacionit. Shumë shpesh te përdoruesit e internetit hapen faqe të vogla në të cilat pa dashje, ose thjesht për kureshtje klikohet duke i dhënë një ndërhyrësi të paautorizuar akses në informacione sensitive dhe personale. Në mënjanimin e këtyre rreziqeve, teknika të tilla si filtruesi aktiv i përmbajtjes, parandalimi i rrjedhjes së informacionit, diktimi i dobësive të aplikimeve ueb janë krijuar dhe aktivizuar. (Benusi, 2015)

Një nga klasat e sulmeve në SaaS etiketohen si Zotri X, apo Man in the Middle attacks (MITM). Në këto sulme një ndërhyrës futet në mes të komunikimit midis dy përdoruesve duke u dërguar atyre informacione pa vlerë, por që i nevojiten atij për të kuptuar dhe deshifruar informacione të ndjeshme. Përdorimi i nënshkrimeve dixhitale në kriptografinë me çelësa publikë dhe zgjedhja me kujdes e algoritmeve të forta kriptografike, bën të mundur parandalimin e sulmeve ndaj programeve të tilla si: Dssniff, Cain, EtterCap, AirJack etj.

Siguria në nivele të ndryshme është e nevojshme për të siguruar një implementim korrekt në retë kompjuterike si: siguria në qasjen e serverave, siguria në internet, siguria e bazës së të dhënave, siguria në fshehtësinë e informacionit, siguria në nivel rrjeti kompjuterik,

siguria e të dhënave në shtresën fizike dhe në atë të aplikimeve, siguria në qasjen e programeve. (Benusi, 2015)

Çështjet etike të Kompjuterizimit në Re

Kompjuterizimi në Re është i bazuar në një ndryshim dhe elementet kryesore janë: Kontrolli është braktisur (refuzohet) nga shërbimet e palës së tretë, të dhënat ruhen në shumë faqe të administruara nga disa organizata dhe shërbime të shumta që bashkëveprojnë në të gjithë rrjetin. Dështimi i infrastrukturës, korruptimi i të dhënave, qasje të paautorizuar apo jashtë shërbimit janë disa nga rreziqet që lidhen me refuzimin e kontrollit ndaj shërbimeve nga palët e treta. Përveç kësaj, sa herë që kemi ndonjë problem është e vështirë për të identifikuar burimin që po e shkakton atë. Struktura komplekse e shërbimeve cloud mund ta bëjë të vështirë për të përcaktuar se kush është përgjegjës në rast se ndodh diçka e padëshirueshme. Ofruesit e shërbimeve cloud kanë grumbulluar informata të ndjeshme personale, të ruajtura në qendrat e të dhënave. Për këtë arsye miratimi dhe pranimi i cloud computing do të përcaktohet nga çështjet e privatësisë të drejtuara nga këto kompani dhe nga vendet ku qendrat e të dhënave janë të vendosura. Intimiteti (privatësia) është prekur nga ndryshimet kulturore ku disa favorizojnë jetën private dhe kulturat e tjera theksojnë komunitetin dhe kjo të çon në një qëndrim të ndarë në dy pjesë ndaj privatësisë në Internet, që është një sistem global. (Marinescu, 2013)

3.6. Kompjuterizimi në Re mobil (Mobile Cloud Computing MCC)

Mobile Cloud Computing (MCC) është një koncept i pranuar gjerësisht që ka për qëllim të përdorë teknikat e cloud computing për ruajtjen dhe përpunimin e të dhënave për pajisjet mobile, duke bërë të mundur uljen e kufizimeve të tyre. (Mobile Cloud Computing, 2015)

Përfitimet që sjellë MCC

- Kompanitë dhe përdoruesit mund të ndajnë burime e aplikacione pa një kosto të lartë harduerike dhe softuerike. Falë kësaj karakteristike bën që pajisjet mobile të kenë një kosto të ulët për përdoruesit fundorë.
- Përfitim kanë dhe zhvilluesit e softuerëve për aplikacione mobile. Përfitimi më i madh është numri i lartë i përdoruesve mobile, gjithashtu një përfitim tjetër është mënyra e funksionimit të Cloud Computing.
- Duke qenë se funksionon me brouser bën që shërbimet të mos ndikohen shumë nga lloji i pajisjeve mobile.

(Mobile Cloud Computing, 2015)

Pikat e dobëta të MCC-së

Edhe pse përfitimi që sjell Cloud Computing në Mobile është shumë i madh ende ka disa çështje që janë për t'u zgjidhur, disa prej të cilave janë (Mobile Cloud Computing, 2015):

- varësia e vazhdueshme nga Interneti,
- mënyra e ndarjes së informacionit,

- siguria e këtij informacioni,
- varësia nga një cilësi e lartë e Internetit (gjë që nuk është gjithmonë e disponueshme në pajisjet mobile).

Arkitektura e MCC-së

Përdoruesit: smartphone, tablet, celular, telefon satelitor

Pranuesit e sinjalit: satelit, AP, BTS

Rrjeti i brendshëm: processori qendror, Serverat, DB, AAA, HA

Fundoret: cloud servers

(Mobile Cloud Computing, 2015)

Arkitektura e aplikacioneve MCC

Sunc: Ky shërbim sinkronizon të gjitha ndryshimet e bëra në pajisjen mobile apo në aplikacionin e pajisjes mobile me shërbimin cloud.

Push: Kjo menaxhon të gjitha ndryshimet e gjendjes duke i çuar këto ndryshime në formë notifikimi përdoruesit në mobile, duke rritur dhe përmirësuar cilësinë e shërbimit për këta përdorues, kështu përdoruesit nuk duhet të kontrollojnë për përditësimet manualisht.

OfflineApp: Ky është një shërbim që ka aftësi të menaxhojë një koordinim midis shërbimeve të ashtuquajtura të nivelit të ulët si sync, push. Kjo u siguron një avantazh të madh zhvilluesve duke hequr përgjegjësinë e shkrimit të kodit për menaxhimin e sinkronizimit të aplikacioneve të tyre me shërbimet cloud. Në momentin që kanali i

komunikimit me Internetin, sigurohet në pajisjen mobile, atëherë të gjitha sinkronizimet dhe notifikimet fillojnë procesin i cili menaxhohet nga OfflineApp.

Network: menaxhon kanalin e komunikimit të nevojshëm për të marrë njoftimet push nga serveri. Kjo ka aftësinë për t'u lidhur me rrjetin në mënyrë automatike. Është një shërbim i nivelit të ulët dhe inkapsulon të gjitha lidhjet që sigurohen me Internetin dhe protokollet e sigurisë.

Database: menaxhon të dhënat lokale të pajisjes mobile. Në varësi të platformës ajo përdor teknika të ndryshme të grumbullimit të të dhënave. Ajo duhet të sigurojë ruajtjen dhe qasjen e të dhënave nga aplikacione të ndryshme duke përdorur teknikën thread-safe. Ashtu si shërbimi Network edhe ky shërbim është i nivelit të ulët.

InterAppBus: Ky shërbim ofron koordinimin e nivelit të ulët dhe komunikimin midis aplikacioneve të ndryshme të instaluara në pajisjen mobile. (Mobile Cloud Computing, 2015)

Shërbimet tipike që nevojiten nga një server MCC

Sync: Shërbimet sync të serverit sinkronizojnë aplikacionet e pajisjeve mobile me shërbimet nga të cilat e kanë origjinën këto të dhëna. Ky proces ndodh për çdo ndryshim gjendje të aplikacionit.

Push: Shërbimet push të serverit monitorojnë të dhënat e kanaleve (nga përdoruesit fundorë) për update që mund të ndodhin. Në momentin që updatet janë gjetur në atë moment, pajisjes mobile i çohet një notifikim për këtë update. Në qoftë se pajisja është jashtë shërbimit d. m. th.: kur s'arrihet për shkak të mungesës së sinjalit atëherë ky njoftim

(notification) pret në një radhë dhe në momentin që kjo pajisje lidhet me rrjetin atëherë të gjitha updatet që janë në radhë i çohen pajisjes.

Secure Socket-Based Data Service: Bazuar në sigurinë që i nevojitet aplikacionit ky shërbim i serverit lejon komunikimin me anë të socket apo me anë të SSL ose në disa raste edhe të dyja.

Security: Komponentët e sigurisë ofrojnë shërbime të autentifikimit dhe të autorizimit të cilat bëjnë të mundur që pajisjet mobile që lidhen në cloud ta kenë të lejuar që t'i qasen sistemit. Para çdo lidhje pajisja mobile duhet të regjistrohet dhe pasi kjo procedurë të kryhet me sukses dhe pajisja mobile të aktivizohet, atëherë kjo ka qasje në shërbimet cloud të kërkuara.

Management Console: Çdo instancë e serverave Cloud duhet të ketë një aplikacion terminal (command line) siç është në këtë rast Management Console e cila i ofron përdoruesve dhe pajisjeve funksionalitet. Në të ardhmen ky shërbim mendohet të ketë karakteristika si përcjellja (tracking), bllokimi etj. (Mobile Cloud Computing, 2015)

Përparësitë e MCC-së

Zgjatja e jetës së baterisë

Janë propozuar disa zgjidhje për të përmirësuar performancën e CPU-së, si dhe për të menaxhuar diskun dhe ekranin në mënyrë inteligjente, duke bërë të mundur reduktimin e konsumit të energjisë. Megjithatë, këto zgjidhje kërkojnë ndryshime në strukturën e pajisjeve mobile, ose kërkojnë një pajisje të re që rezulton në një rritje të kostos dhe nuk mund të jetë e mundshme për të gjitha pajisjet mobile.

Llogaritje teknike “offloading” është propozuar me objektivin për të migruar llogaritjet e mëdha dhe ato të përpunimit të ndërlikuar nga pajisjet me burime të kufizuara (d. m. th.: pajisjet mobile) në makina remote (d. m. th.: servera në rrjetë).

Sipas testeve:

45% e konsumit të energjisë mund të reduktohet për llogaritje të matricave të mëdha. Përveç kësaj shumë aplikacione mobile avancohen nga migrimi i proceseve.

Një optimizues për përpunimin e imazheve mund të zvogëlojë me 41% konsumin e energjisë së një pajisje mobile.

Gjithashtu migrimi i llogaritjeve të lojërave kompjuterike si shahu ul konsumin me 45%. (Mobile Cloud Computing, 2015)

Përmirësimi i ruajtjes së të dhënave

Kapaciteti i ruajtjes është gjithashtu një pengesë për pajisjet mobile. MCC është zhvilluar për t’i mundësuar përdoruesve të pajisjeve mobile ruajtjen e të dhënave dhe qasjen në të dhënat e mëdha në cloud nëpërmjet rrjeteve wireless.

Një shembull tjetër është Exchange Image, e cila përdor hapësirë të madhe për magazinimin në cloud për përdoruesit e pajisjeve mobile. Ky shërbim mobile për ndarjen e fotove online i mundëson përdoruesve mobile që të ngarkojnë imazhe në cloud menjëherë pas shkrepjes. Përdoruesit mund t’i qasen imazheve nga pajisje të ndryshme, qofshin smartphone, tablet, notebook apo desktop; ky imazh i vjen përdoruesit i përpunuar në varësi të pajisjes që është duke përdorur. (Mobile Cloud Computing, 2015)

Figura 8: Ruajtja e të dhënave ne Cloud

Figura 9: Shembuj nga Cloud

Rritja e besueshmërisë

Ruajtja e të dhënave ose e aplikacioneve në ekzekutim e sipër në cloud është një mënyrë efektive për të përmirësuar besueshmërinë e të dhënave dhe të aplikacioneve sepse janë të ruajtura në formë backup në një numër të konsiderueshëm makinash. Kjo çon deri në zhdukjen e mundësisë që të dhënat të humbin në pajisjet mobile. (Mobile Cloud Computing, 2015)

Aplikimet e Mobile Cloud Computing

Mobile Commerce (Tregtia mobile)

Është një plasim aplikimesh apo shërbimesh të cilat janë duke u bërë të arritshme nga pajisjet mobile përmes Internetit. M-Commerce përfshin teknologji të reja, shërbime dhe modele të biznesit. Kjo tregti është mjaft e ndryshme nga e-Commerce tradicionale. Telefonat celularë imponojnë kufizime shumë të ndryshme në krahasim me kompjuterët. Por ata gjithashtu hapën derën e plasimit të aplikimeve dhe shërbimeve të reja. (Sadeh, 2010)

Telefonat celularë i dinë numrat e telefonit të miqve dhe kolegëve tanë. Ata kanë filluar për të gjetur vendndodhjen tonë. Nesër, ata do të zëvendësojnë portofolat tonë dhe kartat e kreditit. Një ditë, ata shumë mirë mund të kthehet në asistentë inteligjentë të aftë për të parashikuar automatikisht shumë nga dëshirat dhe nevojat tona, siç janë: taksi që do të vijnë dhe të na marrin pas takimeve të biznesit ose do të bëjnë përmbledhjen e lajmeve përkatëse dhe të mesazheve të lëna nga kolegët tanë. Por, për të gjitha këto ndryshime që ndodhin, çështjet kryesore të ndërveprimit, përdorshmërisë, sigurisë dhe privatësisë ende duhet të adresohen. (Sadeh, 2010)

Aplikacionet m-commerce mund të klasifikohen në disa klasa duke përfshirë financat, reklamat dhe blerjet. (Sadeh, 2010)

Figura 10: Tregtia mobile

Mobile Learning (Të mësuarit mobil)

Duke përdorur pajisje portative (të tilla si iPad, laptop, PC tabletë, PDAs dhe telefonat e mençur), me rrjetet pa tel (wireless) mundëson lëvizshmërinë dhe të mësuarit në lëvizje, duke i lejuar mësimdhënies dhe të mësuarit që të shtrihet në hapësirat përtej klasës tradicionale. Brenda klasës, të mësuarit mobil i jep instruktorëve dhe nxënësve të rritur fleksibilitet dhe mundësi të reja për ndërveprim. (EDUCUASE|Library)

Të Mësuarit Mobil është hartuar në bazë të të nxënit elektronik (e-learning). Megjithatë, aplikacionet tradicionale kanë kufizime në aspektin e kostos së lartë të pajisjeve dhe rrjetit, kursit të ulët të transmetimit të rrjetit, si dhe burime të kufizuara arsimore. Aplikacionet e të mësuarit, të bazuara në cloud, janë futur për të zgjidhur këto kufizime. (Dharmale & Ramteke, 2015)

Figura 11: Të mësuarit mobil

Mobile Healthcare (Kujdesi shëndetësor mobil)

Shëndeti mobil përdor teknologjinë mobile si mjete dhe platforma për hulumtime shëndetësore dhe ofrimin e kujdesit shëndetësor. Edhe pse telefonat mobil dhe teknologjitë e reja janë përdorur gjithnjë në kërkimin dhe kujdesin shëndetësor, ka të dhëna të kufizuara në dispozicion për të përcaktuar ndikimin e tyre. (Mobile Health)

Qëllimi i aplikimit MCC në aplikimet mjekësore është për të minimizuar kufizimet tradicionale të trajtimit mjekësor (p.sh.: sigurinë, privatësinë dhe gabimet mjekësore). Kujdesi shëndetësor Mobile (m-healthcare) i siguron përdoruesit mobil qasje të përshtatshme në burimet (p.sh.: të dhënat shëndetësore të pacientit) në mënyrë të lehtë dhe të shpejtë. Përveç kësaj kjo i ofron spitaleve dhe organizatave të kujdesit shëndetësor një shumëllojshmëri të shërbimeve online dhe jo të mbajë aplikacionet në serverat lokalë. (Dharmale & Ramteke, 2015)

Figura 12: Kujdesi shëndetësor mobil

Lojërat mobile (Mobile Games)

Lojërat Mobile (m-game) është një nga aplikimet me të ardhura më të larta për ofruesit e shërbimeve. M-Game mund të shkarkojë plotësisht njësimet renderuese dhe përpunuese nga serveri cloud-it, dhe lojtarët vetëm bashkëveprojnë me ndërfaqen e ekranit në pajisjet e tyre. Teknikat që zbatohen në rastin e lojërave mobile, që përdorin cloud, tregojnë fuqinë dhe përfitimet që sjell përpunimi i informacionit në cloud duke bërë të mundur zgjatjen e jetës së baterisë dhe rritjen e performancës së lojërave mobile. (Dharmale & Ramteke, 2015)

Figura 13: Lojëra mobile

Zbatime të tjera praktike

Cloud-i bëhet një mjet i dobishëm për të ndihmuar përdoruesit mobil të ndajnë foto dhe videoklipe në mënyrë efikase dhe të tagojnë miqtë e tyre në rrjetet sociale si Twitter dhe Facebook. MeLog është një aplikacion MCC që u mundëson përdoruesve mobile të ndajnë në kohë reale përvojën e tyre (p.sh. të udhëtimit, të bërit Pazar etj). (Mobile Cloud Computing, 2015)

3.7. Përparësitë e SOA-së

Përparësitë e SOA-së shihen në dy këndvështrime: në IT dhe në biznes.

1. Duke e parë nga pozicioni i IT-së, ajo lehtëson menaxhimin e burimeve të shumta të shpërndara përmes shumë platformave, të cilat kërkojnë më pak investime në harduer dhe janë më të besueshme e më pak të kushtueshme. Kurse, duke e parë SOA-në nga pikëpamja e biznesit, ajo lejon zhvillimin e një gjenerate të re aplikacionesh dinamike që janë zgjidhja e shumë problemeve të biznesit.

2. SOA krijon lidhje të forta midis klientëve dhe ofruesve të shërbimeve. Duke i bërë aplikacionet dinamike dhe shërbimet e bizneseve të përdorshme për përdoruesit e jashtëm dhe ofruesit, ajo arrin jo vetëm një bashkëpunim më të mirë, por rrit në këtë mënyrë edhe kënaqësinë e bashkëveprimit.

3. SOA duke ndihmuar në rritjen e nivelit të informimit për aplikacionet, si dhe rritjen e qasjes së shërbimeve të biznesit, arrin në këtë mënyrë që të dyja palët t'i kuptojnë më mirë kërkesat që kanë ndaj njëri-tjetrit. (Fejzaj, 2014)

3.8. Sfidat e SOA-së

Gjithmonë zhvillimet kanë edhe mangësitë e tyre. Edhe tek SOA të gjitha këto zhvillime shoqërohen nga ana tjetër edhe me disa mangësi. Një prej tyre është edhe siguria, e cila është shumë e rëndësishme në fushën e teknologjisë, mungesa e së cilës sjell vjedhjen e të dhënave të ndjeshme. Prandaj është i nevojshëm validimi i të dhënave dhe rritja e masave të sigurisë. Marrja e masave të nevojshme ndalon sulmet e shumta dhe rrit performancën duke reduktuar duplikimin e kodit. Rritja eksponenciale e numrit të lidhjeve shkakton uljen e shpejtësisë së komunikimit. Menaxhimi i numrit të madh të lidhjeve do të reduktojë numrin e serverave, uljen e kostos, reduktimin e kompleksitetit dhe rritjen e performancës. Përdorimi i memories kashe dhe hostimi i shërbimeve të përbashkëta është një mundësi që duhet aplikuar. (Fejzaj, 2014)

3.9. Përmbledhje

Në këtë kapitull u cekën problemet që patën arkitekturat para SOA-së në lidhje me përshkallëzimin dhe disponueshmërinë e sistemeve IT.

Nga ky kapitull mësuam që SOA është një stil arkitekturor, që udhëheq të gjitha aspektet e krijimit dhe përdorimit të proceseve të biznesit të paketuara si shërbime. Po ashtu në këtë kapitull u sqarua që SOA përcakton dhe siguron që infrastruktura IT të lejojë aplikime të ndryshme që të shkëmbejnë të dhëna dhe të marrin pjesë në proceset e biznesit pavarësisht nga sistemet operative ose nga gjuhët e programimit të këtyre aplikimeve.

Komponentët bazë të një arkitekture të orientuar nga shërbimet janë: ofruesi i shërbimeve, konsumuesi i shërbimeve dhe depozituesi i shërbimeve. Si çdo teknologji e re ashtu edhe SOA ka përparësitë dhe sfidat e veta.

Po ashtu në këtë kapitull theks i veçantë i vihet edhe ueb shërbimeve si një teknologji e re që mundëson qasjen e funksioneve në distancë përmes Internetit.

Për dallim nga arkitektura e SOA-së, arkitektura e ueb shërbimeve është e ndërtuar në pesë shtresa kryesore, të cilat janë të vendosura njëra mbi tjetrën: Discovery, Description, Packaging, Transport dhe Network.

Ueb shërbimet janë të bazuara në një bashkësi XML-standardesh, të cilat janë WSDL (Web Services Descritpion Language), SOAP (Simple Object Access Protocol) si dhe UDDI (Universal Descripption, Discovery and Integration).

HTTP është një standard që paraprin hyrjen e ueb shërbimeve, i cili është zhvilluar për ta lehtësuar transferimin e kërkesave prej brouserit në ueb server. Trafiku i HTTP –së është jashtëzakonisht i mirë, shumica e mureve mbrojtëse e lejojnë trafikun pa ndonjë problem dhe pa ndonjë konfiguracion të veçantë.

Shkëmbimi i SOAP mesazheve dhe WSDL dokumenteve prej një kompjuteri në kompjuterin tjetër bëhet përmes HTTP protokollit (portit 80). Deri më tani shumica e ueb shërbimeve janë ndërtuar me HTTP. HTTP komunikon përmes TCP/IP protokollit.

Google-i është gjiganti që ofron ueb shërbime të shumta në Cloud në formën RESTful. Këto ueb shërbime janë një përmbledhje e ndërfaqeve HTTP që ofrojnë të dhëna gjeografike për aplikacionet që ndërlidhen me harta. Ato përdoren duke bërë një kërkesë HTTP në një URL të veçantë, ndërsa parametrat (obligativë dhe joobligativë) ia kalojmë si argumente në URL. Zakonisht rezultati i HTTP-kërkesës kthehet si JSON ose XML, për të t'u përpunuar më tutje nga aplikacioni ku integrohen këto shërbime.

Disa nga ueb shërbimet e Google-it, tashmë të mirënjohura për programuesit janë:

1) Google Maps Distance Matrix API është një shërbim i cili ofron distancën dhe kohën e udhëtimit nga një pikë e dhënë deri te destinacioni i dëshiruar. Rezultati i kthyer është i bazuar në rrugën e rekomanduar mes pikës fillestare dhe asaj ku synojmë të shkojmë, llogaritur nga Google Maps API dhe përbëhet nga rreshta të cilët përmbajnë distancën dhe kohëzgjatjen e udhëtimit mes çdo palë pikave të dhëna nga A deri te pika B.
2) Google Maps Directions API është shërbim i cili llogarit drejtimet në mes lokacioneve duke përdorur një HTTP-kërkesë. Mund të kërkohet për drejtimet për disa mënyra të transportit, duke përfshirë transit, vozitje, në këmbë ose çiklizëm.
3) Google Maps API lejon integrimin e hartave të Google-it në ueb aplikacione të ndryshme. Zhvilluesit këtë e bëjnë duke përdorur ndërfaqe të JavaScript ose Flash-

it. Ky shërbim është dizajuar të punojë po ashtu në windows - aplikacione si dhe në pajisje mobile.

4) The Google Maps Geocoding API

   Geocoding është procesi i konvertimit të adresave si: *Rexhep Krasniqi, Prishtinë, Kosovë* në koordinata gjeografike (gjerësi 42.644353 dhe gjatësi gjeografike 21.150527) të cilat mund të përdoren për vendosjen e shënjuesve (markers) në hartë të Google-it.

   Reverse Geocoding është procesi i kundërt i kthimit nga koordinatat gjeografike në një adresë të lexueshme.

5) Google Places API

   Të dhënat lokale për mbarë botën. Të dhënat thirren nga e njëjta bazë e dhënash, të cilën e shfrytëzojnë Google Maps dhe Google+. Përmes këtij shërbimi përdoruesi qaset pothuajse në më tepër se një milion biznese, pika interesi dhe bukuri natyrore, bazë e të dhënave e cila është përditësuar nga përdorues të moderuar dhe vërtetuar nga poseduesit e këtyre bizneseve.

Në këtë kapitull gjithashtu lexuat një krahasim në mes Google Maps dhe Bing Maps, dhe i Google Maps me Open Street Map, si konkurrent të Google-it dhe që të tre ofrojnë shërbime të ngjashme.

Pra, Google Maps API duhet të merret në konsideratë nëse ju dëshironi të përdorni versionin falas dhe atë mbi 125 mijë përdorime në vit. Nëse dëshironi të përdorni versionin

me licencë, atëherë duhet t'i jepni përparësi Bing Maps ndaj Google Maps, meqë shërbimet e Bing Maps janë më fleksibile në përmbushjen e kërkesave në lidhje me projektet specifike.

OpenStreetMap është veçanërisht e përshtatshme për përdorim privat dhe është projektuar në mënyrë interaktive. Ndërsa Google Maps ofrojnë ndërfaqe më të mirë për përdoruesin, por dhe është e përshtatshme për vizualizimin e të dhënave të kompanive.

Prezantimi i qartë, rrugët për këmbësorët, për makina dhe për çiklistët si edhe për transportet publike e bëjnë shërbimin e Google-it kaq të dashur dhe të njohur.

Në këtë kapitull ndër të tjera lexuat edhe në lidhje me Kompjuterizimin në Re si një hap i suksesshëm në zhvillimin e përgjithshëm të internetit dhe modeleve të ndryshme të biznesit, ku nënkuptohet shfrytëzimi si shërbim i burimeve të teknologjisë informative siç janë pajisjet teknologjike (harduer) dhe softuerët. Po ashtu u sqaruan modelet e shërbimit të kësaj teknologjie: softueri si shërbim, platforma si shërbim dhe infrastruktura si shërbim. Efikasiteti, qasja e konsumatorëve, mirëmbajtja e lehtë, ulja e shpenzimeve, pavarësia e vendit nga shërbimi, paraqitja virtuale dhe siguria ishin përparësitë e Kompjuterizimit në Re.

Në këtë kapitull u sqarua po ashtu termi i virtualizimit i cili ndryshon nga cloud computing sepse virtualizimi është softuer që manipulon harduer-in, ndërsa cloud computing i referohet një shërbimi i cili është rezultat i manipulimit.

Në këtë kapitull u bë fjalë në lidhje me Mobile Cloud Computing (MCC) si një koncept i pranuar gjerësisht, që ka për qëllim të përdorë teknikat e cloud computing për ruajtjen dhe përpunimin e të dhënave për pajisjet mobile, duke anashkaluar kufizimet e tyre, si dhe cilat janë aplikimet e MCC-s.

Në kapitullin vijues pason programimi i algoritmit gjenetik si ueb shërbim RESTful me Java, meqë është njëri nga objektivat kyç të kësaj teze doktorale. Fillimisht bëhet një hyrje në algoritmin gjenetik, ku sqarohen disa terma dhe principet e punës së tij, mandej pason implementimi i tij. Këtu shpjegohen përbërja e klasave të tij, por jepet dhe një diagram UML-i ku paraqiten variablat, funksionet dhe raportet mes klasave. Këtu është bërë programimi i formulës së Haversinës në Java, si alternativë për të zëvendësuar ueb shërbimin Google Maps Distance Matrix API, meqë ky i fundit nuk ofrohet për vendet që nuk janë në OKB, siç është edhe Kosova.

KAPITULLI IV: PROGRAMIMI I SISTEMIT TË PROPOZUAR

4.1. Programimi i algoritmit gjenetik si ueb shërbim

Sistemi është programuar duke u bazuar në një arkitekturë tashmë të mirënjohur klient-server. I gjithë algoritmi gjenetik është programuar në gjuhën programuese JAVA dhe i tëri ofrohet si ueb shërbim RESTful në serverin TOMCAT.

I gjithë algoritmi gjenetik është programuar përmes NetBeans.

NetBeans është një IDE-së (**I**ntegrated **D**evelopment **E**nvironment) e përshtatshme për të programuar apo zhvilluar aplikacione desktopi, celulari dhe ueb-i përmes gjuhëve programuese Java, JavaScript, HTML5, PHP, C/C++ etj. Pra, thënë shkurt, NetBeans IDE është falas, me kod burimor të hapur që ka një komunitet përdoruesish dhe zhvilluesish në të gjithë botën. (NetBeans IDE, 2017)

Pra, ai është një editor për përpunimin e kodit burimor, që shkruhet nga programuesi, posedon mjete për bashkimin e komponentëve të ndryshëm të programimit si dhe shërben për gjetjen eventuale të gabimeve të bëra nga programuesi gjatë shkruarjes së kodit burimor (Source Code), për të mënjanuar kështu gabimet në aplikim pas përkthimit (compilation).

TOMCAT

Është po ashtu një softuer me kod burimor (source code) të hapur dhe një zhvillim apo programim i teknologjive: Java Servlet, JavaServer Pages, Java Expression Language dhe Java WebSocket. Të gjitha këto teknologji programohen (zhvillohen) nga Java Community Process. Softueri Apache Tomcat është zhvilluar në mjedis (environment) të hapur dhe është publikuar nën versionin e licencës Apache 2. Projekti i Apache Tomcat ka për qëllim të jetë një bashkëpunim i zhvilluesit më të mirë më mbarë botën. Softueri Apache Tomcat përkrah një gamë të gjerë dhe të larmishme të ueb aplikimeve të industrive dhe organizatave të ndryshme. (Apache Tomcat, 2017)

Siç u cek edhe në kapitullin e tretë, para zbatimit të algoritmit gjenetik për zgjidhjen e këtij problemi, problemi duhet të formulohet matematikisht përmes teorisë së grafeve, mirëpo për të mos i përsëritur fjalët, në vijim do të bëj ndërlidhjen e këtyre termave përmes një tabele:

Ndërlidhja mes TSP-së dhe teorisë së grafeve (Alphred)	
Shënimi në lidhje me problemin TSP	**Formulimi në teorinë e grafeve**
Vendi	Nyja e grafit
Lidhja në mes vendeve	Brinja e grafit
Distanca, Kostoja	Brinja e vlerësuar e grafit
Bashkësia e vendeve dhe lidhjeve	G=(V, E); Grafi=(Bashkësia e nyjeve, Bashkësia e brinjëve)
Struktura e përbërë nga vendet dhe lidhjet (me distanca të dhëna) pa drejtim të orientuar.	Graf i ponderuar dhe i paorientuar
Nëse ekzistojnë lidhje në mes të gjithë	Graf i plotë

vendeve.	
Udhëtimi	Nënbashkësi e brinjëve të grafit
Turi	Rreth i Hamiltonit
Problemi i biznesmenit udhëtar.	Kërkimi i rrethit të Hamiltonit në grafin e plotë.

Tabela 14: Ndërlidhje konceptesh mes TSP-së dhe teorisë së grafeve

Simulimi i evolucionit në kompjuter

Meqë përmes algoritmit gjenetik programuesi mundohet të simulojë një pjesë të natyrës në kompjuter (evolucionin- ku më i forti mbijeton), atëherë programimi i këtij algoritmi duhet të simulojë këto faza: gjenerimin e popullatës, përzgjedhjen në mënyrë të rastësishme, bashkëdyzimin, ndryshimin e rastësishëm në gjene, vlerësimin si dhe hedhjen apo zëvendësimin e individ të ri mes dy individëve të përzgjedhur më parë.

Këto faza sqarohen grafikisht në figurën e mëposhtme (Häckel & Lemke, 2012):

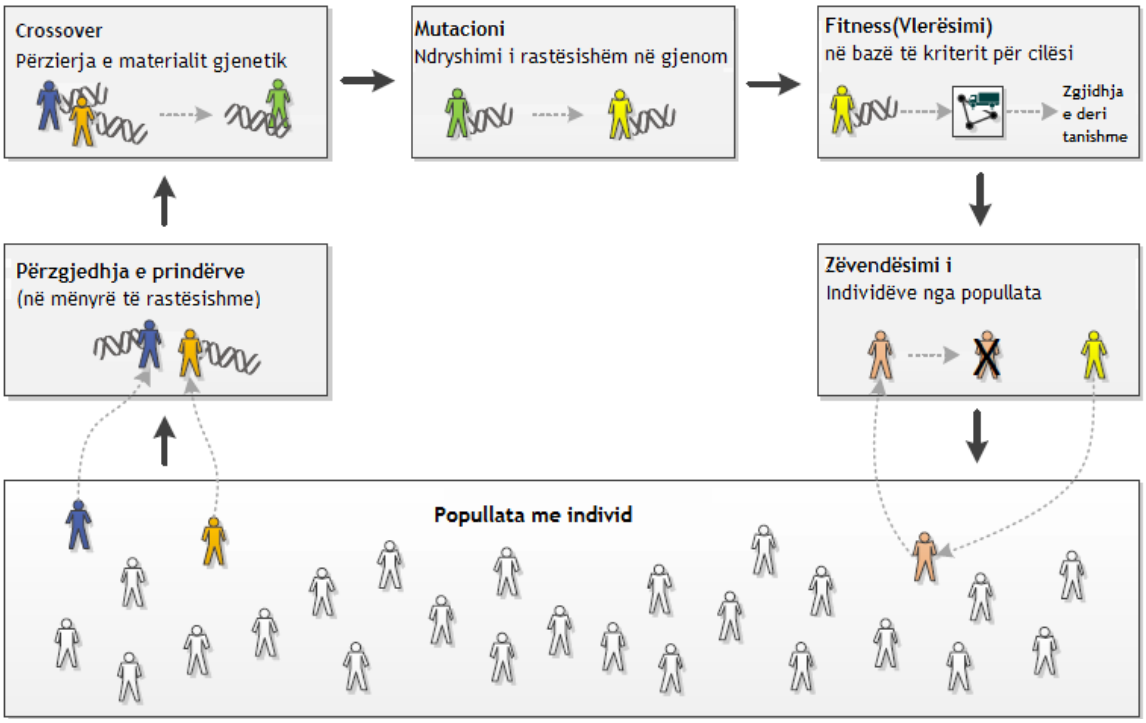


Figura 14: Principi i punës së algoritmit gjenetik

Siç shihet në figurën e mësipërme, evolucioni ka të bëjë me gjenetikën nga biologjia dhe për këtë arsye e shoh të pashmangshme sqarimin dhe ndërlidhjen e disa termave biologjikë, ndërlidhjen dhe domethënien e tyre tek algoritmi gjenetik. Pra, në tabelën e mëposhtme sqarohen disa terma në mënyrë që algoritmi gjenetik dhe puna e tij të kuptohet drejt.

Ndërlidhja e termave biologjikë dhe atyre të algoritmit gjenetik (Häckel & Lemke, 2012)	
Termi biologjik	**Domethënia tek algoritmi gjenetik**
Individi	Përfaqëson në mënyrë të përshtatshme zgjidhjen e mundshme të problemit.
Kromozomi	Zakonisht identik me konceptin e individit.
Popullsia (Popullata)	Bashkësia e individëve (bashkësia e zgjidhjeve të mundshme).
Fitness	Cilësia e zgjidhjes në lidhje me kriteret e synuara (funksion vlerësimi).

Prindërit	Individë të përzgjedhur për riprodhim (Me anë të përzgjedhjes së varur nga Fitness).

Tabela 15: Domethënia e termave biologjikë tek algoritmi biologjik

Zbatimi i algoritmit gjenetik në zgjidhjen e problemit TSP

1) Krijimi i tureve në mënyrë të rastësishme të ashtuquajtura popullatë. Ky algoritëm i jep përparësi apo lidh qytetet që janë të afërt me njëri-tjetrin. Popullata është një nënbashkësi turesh nga bashkësia e gjitha zgjidhjeve të mundshme. Secili individ, që përbën popullatën, shënjohet me gjasë të njëjtë për t'u përzgjedhur dhe të bëhet prind. Përngjason me marrjen e një mostre për eksperiment nga një specie e caktuar.

2) Ky algoritëm përzgjedh dy ture në mënyrë të rastësishme dhe i bashkon ato në një të vetme, me shpresë që turi i krijuar do të jetë më i shkurtër. (Imbach)

3) Në turin e posakrijuar bëhet një ndryshim i rastësishëm (ndryshim në renditjen e vendeve apo qyteteve për t'u vizituar). Përngjason me ndryshimin e rastësishëm në gjene të ndonjë individi qoftë nga kushtet natyrore apo nga ndonjë eksperiment laboratori, dhe pritet që individi të ketë anomali apo ndonjë përmirësim. (Imbach)

4) Përmes funksionit për vlerësim bëhet përllogaritja e gjatësisë së turit dhe vlerësimi i tij: nëse është i përdorshëm apo jo. Kjo ngjan me vlerësimin e eksperimentit gjenetik duke e vlerësuar në bazë të kritereve të caktuara. (Imbach)

5) Nëse përmes funksionit për vlerësim del që ky tur është i rëndësishëm, atëherë bëhet zëvendësimi i dy tureve të përzgjedhura në mënyrë të rastësishme, me turin e ri. Pra,

popullata e tanishme përmban turin e ri. Nëse turi i ri vlerësohet nga funksioni për vlerësim si i papërdorshëm, meqë gjatësia e tij nuk kontribuon në shkurtimin e gjatësisë së turit, atëherë flaket tutje dhe në këtë mënyrë bëhet pastrimi i popullatës. Pra, në këtë mënyre zgjidhjet më të mira dominojnë dhe kështu bëhet pastrimi i popullatës epokë pas epoke apo cikël pas cikli. Meqë zgjidhjet më të mira ruhen për faza të mëtejshme, ka analogji me natyrën ku më i forti mbijeton. (Imbach)

Faza 2 deri në 5 quhet cikël, brez ose epokë.

Parametrat që kontrollojnë punën e algoritmit gjenetik

1) Madhësia e popullatës – Madhësia e popullsisë është numri fillestar i tureve të gjeneruara në mënyrë të rastësishme që janë krijuar kur algoritmi fillon punën. Një popullsi e madhe zgjat më shumë për të gjetur një rezultat. Një popullsi më e vogël rrit mundësinë që çdo tur në të do të duket përafërsisht i njëjtë. Kjo rrit mundësinë që zgjidhja më e mirë nuk do të gjendet, meqë zgjidhjet më të mira nuk do ta kenë fatin të gjenerohen. (Vishnupriyan, Govindarajan, Prabhakaran, & Ramachandran)
2) Fqinjësia/madhësia e grupit – Në çdo gjeneratë, përzgjedhja e tureve bëhet në mënyrë të rastësishme, nga të cilët dy më të mirët quhen prindërit. Numri i madh ndikon në rritjen e gjasës që turet të mira të përzgjidhen si prindër, mirëpo në anën tjetër kjo bën që shumë ture të mos përzgjidhen (të mos përfillen) nga algoritmi. (Vishnupriyan, Govindarajan, Prabhakaran, & Ramachandran)
3) Përqindja e mutacionit – Është përqindja që çdo fëmijë pas bashkëdyzimit do të pësojë mutacion. Kur një tur ka pësuar mutacion, kjo do të thotë që një vend, në vargun e

vendeve për t'u vizituar, është zhvendosur nga një pikë në pikën tjetër. (Vishnupriyan, Govindarajan, Prabhakaran, & Ramachandran)

4) Numri i vendeve fqinjë- algoritmi gjenetik ka prirje që vendet për t'u vizituar, që gjenden afër njëri tjetrit, t'i lidhë dhe nga ato të formojë turet fillestare. Kur të gjenerohet popullata fillestare, atëherë ky është numri i vendeve që shikohen nga algoritmi si vende fqinje.
5) Numri maksimal i brezave – sa breza duhet të kalojnë në mënyrë që algoritmi ta ndërpresë punën e tij apo të pushojë së ekzekutuari. (Vishnupriyan, Govindarajan, Prabhakaran, & Ramachandran)

Implementimi i algoritmit gjenetik (përbërja dhe shpjegimi i klasave)

Vendi

Klasa Vendi është struktura themelore e të dhënave ku ruhen shënimet dhe atributet e një vendi në hartën e Google-it. Pra, është struktura bazë e të gjitha klasave të jera si dhe e vetë algoritmit gjenetik. Vetitë e një vendi, që ruhen dhe manipulohen përmes kësaj klase, janë: ID-ja unike e vendit në hartë të Google-it (ID-ja e vendit në serverët e Google-it), emri i vendit, adresa si dhe gjerësia dhe gjatësia gjeografike e tij. (Latitude/Longitude)

Para se algoritmi të fillojë punën e tij është e domosdoshme të dihen distancat mes vendeve për t'u vizituar. Distancat nuk dihen që nga fillimi sepse përdoruesi e ka në dorë zgjedhjen e vendeve që do të vizitojë përmes aplikacionit të Android-it në formë hartash të Google-it. Pra, përdoruesi shënjon në hartë vendet për vizitë dhe distanca në mes tyre nuk dihet.

Google-i ofron një ueb shërbim (Google Maps Distance Matrix API) përmes të cilit mund të gjendet distanca mes dy apo më shumë vendeve në harta të Google-it, mirëpo ky shërbim nuk është në dispozicion për të gjitha ato vende që nuk janë vende anëtare të OKB-së, siç është Kosova. Pra, ky lloj implementimi i funksionit për vlerësim bën që algoritmi gjenetik të jetë i kufizuar për disa vende dhe më i ngadalshëm gjatë punës së tij.

Alternativa tjetër e implementimit të funksionit për vlerësim është programimi i formulës së Haversinës, implementim ky që bën algoritmin gjenetik më të shpejtë gjatë ekzekutimit, mirëpo nuk është i saktë në vlerësimin e gjatësisë së një turi. Arsyet do të cekën në nënkapitullin përkatës.

Pra, sikur të programoja duke shfrytëzuar vetëm këtë shërbim Google-i, aplikacioni i doktoratës sime do të ishte i kufizuar, do të funksiononte për disa vende e për disa tjera jo. Më është dashur të gjej edhe alternativën tjetër në mënyrë që puna praktike e doktoratës sime (i gjithë sistemi i programuar) të jetë sa më e përgjithshme. Në rastin tim formula e Teoremës së Pitagorës nuk vjen në pyetje, meqë përmes saj mund të gjendet largësia në mes të dy pikave në rrafsh. E vetmja alternative tjetër që kam, në të cilën toka trajtohet si sferë e jo si sipërfaqe, është implementimi i formulës së Haversinës, cila është si më poshtë (Rubin, 2013):

R = radiusi i Tokës (radiusi = 6,371 km)

$\Delta lat = lat2 - lat1$

$\Delta long = long2 - long1$

$a = \sin^2(\Delta lat/2) + \cos(lat1)*\cos(lat2)*\sin^2(\Delta long/2)$

c = 2*atan2(√a, √(1−a))

d = R*c

Programimi i funksionit për vlerësim duke shfrytëzuar formulën e Haversinës

Formula e Haversinës

```
publicstatic double neRadian (double shkalla) {
        return shkalla * Math.PI / 180;
}

publicdouble distanca (double lat1, double lon1, double lat2, double lon2) {
        double radiusi_tokes = 6371.00;
        double distanca_gjeresise = neRadian (lat1 - lat2);
        double distance_gjatesise = neRadian (lon1 - lon2);
        double lat1_r = neRadian (lat1);
        double lat2_r = neRadian (lat2);
        double a = Math.pow(Math.sin( distanca_gjeresise / 2 ), 2) +
                Math.pow(Math.sin( distanca_gjeresise / 2 ), 2) *
         Math.cos(lat1_r) * Math.cos(lat2_r);
     double c = 2.0 * Math.atan2(Math.sqrt(a), Math.sqrt(1.0 - a));
     return radiusi_tokes * c;
}
```

Tabela 16:Programimi i formulës së Haversinës

Siç u përmend edhe më lart, ky lloj implementimi i funksionit për vlerësim e bën algoritmin gjenetik më të përgjithshëm dhe më të shpejtë gjatë ekzekutimit. Pra, më të përgjithshëm për shkak se gjatësia e një turi matet duke zbatuar formulën e Haversinës për të matur distancën në mes të çdo dy vendeve të turit. Kjo ndodh shumë shpejt meqë formula është e programuar lokalisht në server pa pasur nevojë të konsumohet ndonjë ueb shërbim. Kjo formulë është e përgjithshme dhe vlen për të gjitha vendet e globit të Tokës, pa ndonjë kufizim. Këtu Toka trajtohet si sferë dhe jo si gjeoid i valëzuar, ashtu siç është në të vërtetë, dhe distanca mes dy vendeve matet pa marrë në konsideratë se a ka rrugë që lidh ato dy vende në realitet. Pra, në natyrë mund të mos ketë rrugë që t'i lidhë ato dy vende

drejtpërdrejt, mirëpo aty dy vende lidhen përmes një rruge që kalon përmes një vendi të tretë të ndërmjetëm, ose është kodër apo mal dhe duhet t'i sillemi rrotull. Në këto raste vlerësimi i distancës nuk është i saktë.

Mendim kritik në lidhje me përdorimin e kësaj formule

Përdorimi apo implementimi i kësaj formule edhe pse përshpejton punën e algoritmit në krahasim me konsumimin e ueb shërbimit të Google-it, ka në vetvete dy të meta:

a) Këtu Toka trajtohet si sferë e jo si gjeoid i valëzuar, ashtu siç është në të vërtetë, dhe si rrjedhojë e kësaj distanca në mes vendeve, që gjenden afër ekuatorit apo në mes vendeve që gjenden afër poleve, mund të nënvlerësohet ose të mbivlerësohet.
b) Distanca mes vendeve llogaritet pa marrë parasysh situatën reale në terren. Kjo nënkupton që dy vende në terren mund të mos kenë lidhje të drejtpërdrejtë, por të lidhen përmes një rruge, përmes një vendi të tretë (të ndërmjetëm apo rreth e rrotull), dhe si rrjedhojë e kësaj, rruga në të vërtetë është më e gjatë. Në raste të tilla gjatësia e rrugës përmes kësaj formule llogaritet sikur këto vende të kishin një lidhje të drejtpërdrejtë. Pra, thënë shkurt, si rezultat marrim një gjatësi më të shkurtër sesa ajo që është në realitet dhe se gjatësia e turit në fund është më e shkurtër sesa në terren. Meqë distanca në mes të vendeve për algoritmin gjenetik është jetike, bazuar në distancën e llogaritur përmes formulës së Haversinës, ky algoritëm mund të mos marrë vendime të duhura.

Programimi i funksionit për vlerësim duke shfrytëzuar Google Distance Matrix

Kjo formë tjetër e programimit të funksionit për vlerësimin e gjatësisë së turit, është 100% e saktë, meqë konsumimi i ueb shërbimit Google Distance Matrix, jep distancën e saktë të rrugës që lidh dy vende, meqë ky shërbim vlerëson rrugën që lidh realisht ato dy vende e që është e evidentuar në bazën e të dhënave të Google-it. Kjo bën që algoritmi të jetë shumë i saktë në punën e tij, mirëpo i kufizuar vetëm për vendet që janë anëtare të OKB-së, meqë vetë shërbimi Google Distance Matrix ofrohet vetëm për vende që janë anëtare të OKB-së. Përveç mangësisë që algoritmi nuk është i përgjithshëm, mangësia tjetër e këtij lloj programimi është që algoritmi ngadalësohet në përgjithësi. Shkaku i ngadalësimit të punës së algoritmit është konsumimi i këtij ueb shërbimi për vlerësim. Pra, koha e pritjes së kthimit të përgjigjes nga serveri i Google-it si dhe koha e përpunimit (parësimit) të kësaj përgjigjeje janë kritike për algoritmin. Përdorimi i këtij ueb shërbimi bëhet në mënyrë të përsëritur aq herë, sa është numri i gjeneratave të zgjedhura nga përdoruesi si parametër hyrës i algoritmit. Numri i gjeneratave është parametër hyrës, që ndikon punën e algoritmit gjenetik, dhe tregon pas sa gjeneratash ky algoritëm duhet të pushojë së ekzekutuari. Për shembull, nëse përdoruesi zgjedh që algoritmi duhet të pushojë së ekzekutuari pas 30 gjeneratash, kjo nënkupton që në çdo gjeneratë ndodh një ndryshim i rastësishëm në renditjen e vendeve për t'u vizituar dhe 30 herë në mënyrë të përsëritur duhet të vlerësohet gjatësia e turit pas ndryshimit, duke shfrytëzuar këtë ueb shërbim. Sa më i madh të jetë numri i gjeneratave aq më e madhe do të jetë koha e ekzekutimit të algoritmit duke përfshirë kohën e të gjitha operacioneve bazë në përgjithësi (lokalisht), dhe kohën e shfrytëzimit të këtij ueb shërbimi të Google-it në veçanti (në distancë).

Shfrytëzimi i shërbimit Google Distance Matrix

```
public double gjatesiaTurit(String prej, ArrayList<String> deri, String API_KEY) {
   long distanca = 0;
try {
      JsonReader jsonClass = new JsonReader();
      StringBuilder builder = new StringBuilder();
      builder.append("https://maps.googleapis.com/maps/api/distancematrix/json?”);
      builder.append("origins=").append(prej);
      Iterator itr= deri.iterator();
      int i = 0;
      String origins = "";
      while(itr.hasNext()) {
         if (i == deri.size() - 1) {
            origins = origins + itr.next();
         } else {
            origins = origins + itr.next() + "|";
            i++;
         }
      }
      builder.append("&destinations=").append(origins);
      builder.append("&mode=driving");
      builder.append("&units=metric");
      builder.append("&key=").append(API_KEY);
      final JSONObject pergjigja = jsonClass.read(builder.toString());
      JSONArray rezultati = pergjigja.getJSONArray("rows");
      distanca = rezultati.getJSONObject(0).getJSONArray("elements").getJSONObject(0)
.getJSONObject("distance").getInt("value");
   } catch(Exception e) {
   }
   return distanca / 1000;
}
```

Tabela 17: Programimi i funksionit për llogaritjen e gjatësisë së turit duke përdorur Google Distance Matrix

Siç mund të vërehet lehtë, brenda funksionit “gjatesiaTuri” përdoret klasa JsonReader, të cilën e kam programuar vetë, mirëpo për të mos e zgjatur kodin po e lë pa e shpjeguar, meqë do t’ia bashkëngjis disertacionit te shtojcat (shtojca A).

Klasës vendi po ashtu i janë programuar edhe dy funksione ndihmëse:

ktheObjekt – shërben për të kthyer rezultatin si objekt (pra të gjitha atributet e një vendi të hartës së Google-it).

konvertoNëJSON - shërben për të kthyer rezultatin si format JSON-i.

Turi

Në këtë klasë përdor dy struktura të dhënash që ofrohen nga vetë JAVA: *ArrayList*<> dhe *Collections.*

Kjo klasë përmban në veti këto funksione për qasje në tur dhe manipulim me të:

Konstruktori Tur pa parametër hyrës, që shërben për gjenerimin e një turi të zbrazët në rast nevoje, si dhe një konstruktor për gjenerimin e komplet turit.

shtoVend – shton një vend për t'u vizituar në pozicion të caktuar të turit. Pra, pranon dy parametra hyrës: vendi për t'u vizituar si objekt, si dhe pozicioni i caktuar ku dëshirojmë ta fusim vendin e caktuar në tur.

thirrVendin – përmes këtij funksioni thërras vendin me indeks të caktuar nga turi. Pra, ky funksion pranon si parametër hyrës indeksin përmes të cilit kthehet si rezultat vendi me pozicion të caktuar nga turi.

përmbahetVendi – programimi i këtij funksioni ka qenë jetik, meqë përmes këtij verifikohet nëse vendi i caktuar përmbahet në tur apo jo. Pra, në tur nuk duhet të futen dy vende të

njëjta për t'u vizituar. Si parametër hyrës pranon vendin si objekt për të cilin dëshirojmë të sigurohemi se a përmbahet në tur apo jo. Pra, në këtë mënyrë evitojmë futjen e dy vendeve të njëjta në tur.

madhësiaTurit – kthen si rezultat nga sa elemente apo vende për t'u vizituar përbëhet turi aktual. Pra, sa vende për t'u vizituar janë në turin aktual.

llogaritGjatësinë – ky funksion llogarit gjatësinë e komplet turit, i cili përdoret nga funksioni për vlerësim.

Fitness – ky funksion vlerëson turin aktual. Pra, ky funksion jep vlerësimin në gjasë 1/gjatësia totale e turit, sa është gjasa që turi aktual të ripërdoret prapë apo të rikombinohet në fazat e mëtejshme. Sa më e madhe të jetë gjatësia e turit aq më e vogël është gjasa të ripërdoret dhe anasjelltas.

gjeneroTurin – përmes një LOOP-i përpunohen të gjitha vendet që dëshirojmë t'i vizitojmë dhe bëhet futja e tyre në një tur, ku mandej në mënyrë të rastësishme rirenditet turi te Collections.

Popullata

Programimi i kësaj klase ka qenë i domosdoshëm, në mënyre që ta kem një strukturë të dhënash ku i ruaj dhe përmes së cilës menaxhoj me turet kandidate. Pra, është strukturë e të dhënave ku ruhen të gjitha turet kandidate e që nënkupton një nënbashkësi nga bashkësia e të gjitha zgjidhjeve ose tureve të mundshme që duhet të trajtohen për zgjidhjen e problemit.

Si strukturë të dhënash kjo klasë në vetvete përdor array ku si elemente të tij janë objektet e klasës tur.

Kjo klasë ka në vetvete këto funksione publike:

ruajTurin – ku në array-in e tureve shton në pozicionin e caktuar një tur të ri. Pra, si parametra hyrës pranon indeksin për pozicionin e caktuar si dhe turin si objekt.

thirrTurin – thirr turin e caktuar me indeks të caktuar nga array-i. Si parametër hyrës pranon një indeks (numër i plotë), përmes të cilit thirret një tur me indeksin e dëshiruar nga array-i.

thirrMëTëPershtatshmin – thirr turin më të përshtatshëm të vlerësuar nga funksioni për vlerësim dhe që është turi me gjatësinë më të vogël nga popullata.

madhësiaPopullatës – tregon numrin e tureve në popullatën aktuale.

Popullata – konstruktori popullata pranon dy parametra hyrës aq sa duhet të jetë madhësia e popullatës dhe një argument boolean a duhet të gjenerohet ajo me elemente në të dhe që janë turet kandidate.

Klasa Algoritmi gjenetik

Crossover – ky funksion pranon dy parametra hyrës të tipit të klasës Tur. Pra, këta dy parametra (dy turet) janë përzgjedhur në mënyrë të rastësishme dhe përmes këtij funksioni bashkëdyzohen në një tur të vetëm. Në mënyrë të rastësishme caktohet pozicioni fillestar

dhe ai përfundimtar i turit të parë, ku përmes një LOOP-i përpunohen vendet e turit të parë me elementet që gjenden në pozicionet (nga dhe deri) dhe bëhet kopjimi i tyre në një tur ndihmës që do të jetë turi i ri. Ndërsa vendet e turit të dytë përpunohen prapë përmes një LOOP-i dhe me elementet e tij plotësohen pozicionet e ngelura të turit të ri. Pra, si rezultat kemi turin e ri (individin e ri me material gjenetik të përzier nga dy prindërt e përzgjedhur). Numri i elementeve të turit të kombinuar është po ai i njëjti, si i njërit nga prindërit e tij, mirëpo elementet e tij janë kombinuar nga vendet për t'u vizituar prej të dy tureve që shërbyen si prindër.

mutate- ky funksion pranon një parametër hyrës të tipit tur, dhe kthen si rezultat të njëjtin tur me një dallim, por ku është bërë një ndryshim i vetëm në mënyrë të rastësishme në renditjen e vendeve për t'u vizituar. Pra, gjenerohen në mënyrë të rastësishme dy pozicione, dhe elementet përkatëse që i përkasin këtyre dy pozicioneve në tur ndryshojnë vendin e tyre.

turnamentSelection – Përmes këtij funksioni sistemit i mundësohet që të përzgjedhë turin më të përshtatshëm (që konsiderohet njëra palë prindërore), për funksionin crossover. Pra, përzgjidhet turi elitë, duke përpunuar dhe llogaritur fitnessin e të gjitha tureve kandidate nga popullata dhe kështu në bazë të fitnessit më të mirë përzgjidhet turi më i mirë i mundshëm.

evolvePopulation – ky funksion pranon një parametër hyrës të tipit të klasës së popullatës dhe po ashtu kthen si rezultat një popullatë të evoluar për më tepër se një brez. Numrin e brezave e cakton përdoruesi, ndërsa ky funksion përdor që të tria funksionet e tjera për ta

evoluar popullatën e dhënë: crossover, mutate dhe turnamentSelection. Këta hapa do të përsëriten përderisa nuk janë plotësuar kushtet që algoritmi gjenetik të pushojë së vepruari.

Kriteret e ndërprerjes së punës të algoritmit gjenetik

Kriteri i mbarimit cakton pikën kohore kur algoritmi duhet të përfundojë së ekzekutuari. Edhe këtu ka mundësi të ndryshme, prej të cilave vetëm disa vlejnë të përmenden. Pushimi i algoritmit mund të pasojë (Martín, Jessica, Ignacio, & Nélida, 2004):

- Pas një numri të caktuar të gjeneratave, brezash apo epokash (shiko 6.2 nga 2-5).
- Sapo turi më i mirë nuk mund të përmirësohet pas një numri të caktuar gjeneratash.
- Sapo popullata apo një përqindje e saj të përbëhet nga ture identike.
- Pas kalimit të një periudhe kohore.

Diagrami i klasave të algoritmit gjenetik

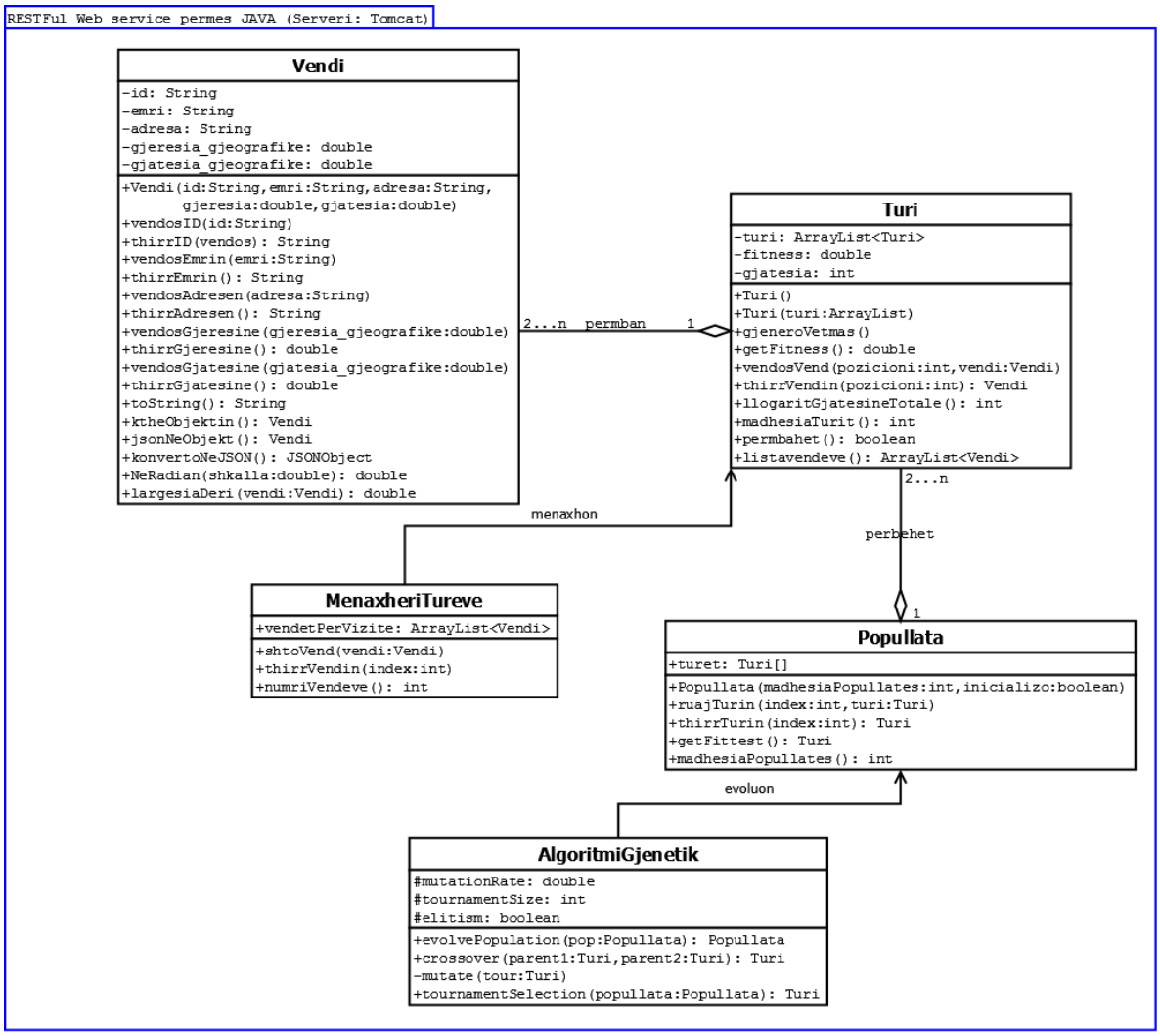


Figura 15: Diagrami i klasave të algoritmit gjenetik

Ofrimi i algoritmit gjenetik si ueb shërbim RESTful (Web-Service)

Një nga objektivat e punimit praktik të doktoraturës sime ka qenë edhe ofrimi i algoritmit gjenetik si ueb shërbim RESTful.

Algoritmin e kam programuar duke u bazuar në arkitekturën REST në gjuhën programuese JAVA duke përdorur JAX-RS. I gjithë algoritmi rri në ekzekutim e sipër në serverin TOCAT, duke pritur kështu ndonjë kërkesë HTTP-Post që t'i kthejë përgjigje (ose t'i ofrojë

shërbim). Aplikacioni server përgjon portin e caktuar dhe pranon të dhëna të formatit JSON. Me të pranuar kërkesën përmes HTTP-Post, ai përpunon formatin JSON, duke ia bërë kështu ato të përshtatshme, përgatit dhe i përcjell ato tek algoritmi gjenetik që ai të kryejë punën e tij. Pasi algoritmi gjenetik ta ketë kryer punën e tij, pra ta ketë optimizuar rrugën, bazuar në koordinatat gjeografike, rezultati i përfituar përsëri paketohet në formatin JSON dhe i kthehet pajisjes me sistem operativ Android e cila dërgoi kërkesën HTTP-Post për optimizim. Pra, ueb-shërbimi RESTful pranon dhe prodhon të dhëna të formatit JSON.

Shkëputje nga kodi i ueb shërbimit RESTful

```
@POST
@Consumes("application/json")
@Produces("application/json")
public String postJson(String content) {
  JSONObject rezultati = new JSONObject();
   try {
      JSONObject json = new JSONObject(content);
      JSONArray jsonArray = json.getJSONArray("vendet");
      int gjatesia = jsonArray.length();
      for(int i=0; i<gjatesia; i++) {
         MenaxheriTureve.shtoVend(
               new Vendi(
                  jsonArray.getJSONObject(i).getString("id"),
                  jsonArray.getJSONObject(i).getString("emri"),
                  jsonArray.getJSONObject(i).getString("adresa"),
                  jsonArray.getJSONObject(i).getDouble("gjeresia_gjeografike"),
                  jsonArray.getJSONObject(i).getDouble("gjatesia_gjeografike")
               )
         );
      }
      Popullata pop = new Popullata(30, true);
      pop = AlgoritmiGjenetik.evolvePopulation(pop);
      for (int i = 0; i < 100; i++) {
         pop = AlgoritmiGjenetik.evolvePopulation(pop);
      }
      ArrayList<Vendi> vendet = pop.getFittest().listavendeve();
      rezultati.put("vendet", arrayListToJSON(vendet));
   } catch(Exception $ex) {
```

```
    }
    return rezultati.toString();
}
```

Tabela 18: Pjesë nga kodi i ueb shërbimit RESTful

Përmes figurës së mëposhtme sqarohet grafikisht komunikimi mes pajisjes me sistem operativ android dhe serverit Tomcat ku ofrohet ueb shërbimi RESTful. Pra, aplikacioni që kam programuar për pajisjet me sistem operativ, mbledh informatat në lidhje me vendet e zgjedhura nga përdoruesi në hartën e Google-it dhe e inicion kërkesën për konsumin e ueb shërbimit.

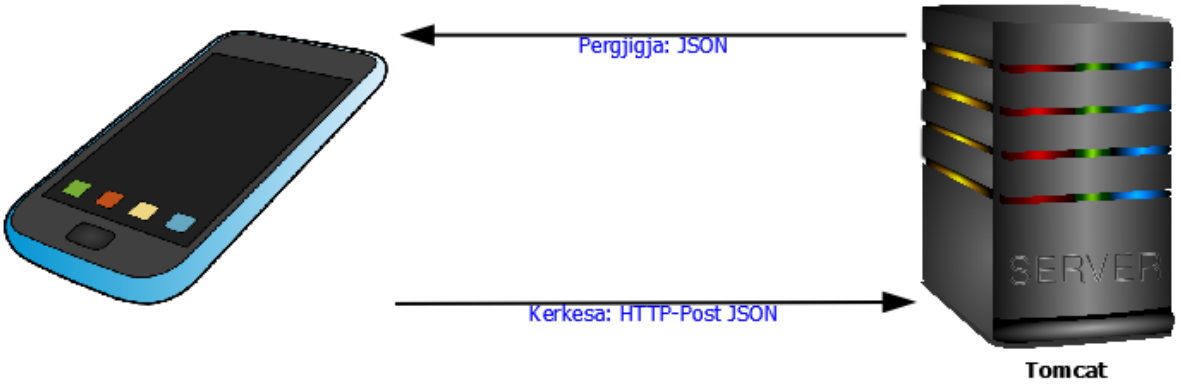


Figura 16: Komunikimi mes pajisjes inteligjente dhe serverit TOMCAT

4.2. Programimi i aplikacionit për pajisje mobile me sistem operativ Android

Hyrje në sistemin operativ Android

Android-i është një sistem operativ mobil i bazuar në sistemin tjetër operativ Linux i cili është iniciuar nga Google dhe shumica e funksioneve të tij janë të lidhura ngusht me shërbimet e kërkimit të këtij gjiganti.

Androidi si sistem operativ mobil me kod burimor të hapur mbështetet nga Open Handset Alliance. Përmes tij përfaqësohen edhe gjigantët e mëdhenj të industrisë siç janë: Intel, Samsung, Sony, Motorola dhe HTC.

Androidi për shkak të popullaritetit të tij zhvillohet gjithnjë e më tepër dhe është në ditët e sotme një sistem i plotë. Tipari i tij kryesor është lidhja e tij me shërbimet e internetit. Ka arritur sukses për shkak të bashkëpunimin të tij me Adobe Flash, ku deri në versionin 4.0 ka pasur përkrahje të rregullt. Në nëntor të vitit 2007, Google plasoi sistemin Android për herë të parë. (Trautmann, 2016)

Ofruesit e telefonave të mençur dhe tabletave të ndryshëm e përshtasin sistemin operativ Android për pajisjet e tyre dhe janë gjithashtu përgjegjës për të ofruar mandej përditësime softueri për pajisjet e tyre. Kjo ndonjëherë shkakton frustrim në mesin e përdoruesve kur përditësime të rralla ofrohen për versionet e fundit të Android-it.

Ky sistem konsiderohet fleksibël dhe i hapur. Google "Play Store" (ish "Android Market") ofron një numër shumë të madh të programeve të tjera. Shumë aplikime mund të instalohen edhe nga burime të tjera. Shumica e telefonave të mençur me sistem operativ Android kanë vend për të futur një kartë memorie shtesë dhe mund të lidhen shumë lehtë përmes USB-Drive ose përmes Media Transfer Protocol (MTP) me kompjuterin, për të

shkëmbyer të dhëna të ndryshme. Sinkronizimi i adresave dhe i kalendarit bëhet përmes Internetit. Disa nga ofruesit e telefonave të mençur ofrojnë softuerin e tyre për sinkronizimin e të dhënave mes kompjuterit dhe telefonit përmes USB-së. (Die Betriebs-systeme im Überblick, 2016)

Android-i është sistem operativ dhe njëkohësisht platformë softueri për pajisjet si: telefona të mençur, laptop-a dhe tableta të ndryshëm. Kohët e fundit është konkurrenti më i madh i platformës së Apple-it IOS. Android-i ofron një numër të madh përparësish kundrejt IOS-it dhe për këtë arsye përdoret nga një numër i madh telefonash të mençur dhe tabletash.

Përparësitë e Android-it

1) Android-i është softuer falas me kod burimor të hapur. Kjo do të thotë që zhvilluesit të softuer-it mund t'i qasen publikisht dhe ta shohin atë. Zhvillimi i mëtejshëm i tij është i hapur për të gjithë.
2) Një tjetër përparësi e rëndësishme e platformës Android është fakti se Google-i nuk është shumë kufizues në trajtimin e tij. Kjo do të thotë që zhvilluesit e pavarur munden shumë shpejt të zhvillojnë aplikime të ndryshme për këtë platformë dhe brenda disa orësh apo ditësh t'i publikojnë ato në PLAY-STORE, ndërkohë që Apple ka një proces të gjatë dhe të mundimshëm pranimi, ku shumica e aplikimeve refuzohen të publikohen.

(Android-Betriebssystem: Vor- und Nachteile, 2016)

Versioni	Plasuar	Karakteristika	Harduer (Telefoni)
1.0	10/2008	/	T-Mobile G1
1.1	02/2009	- Ruajtja e MMS-Attachments - Përkrahje e zgjeruar e Google-Maps	-
1.5 (Cupcake)	04/2009	- Video inçizim dhe rishikim - Stereo përmes Bluetooth - Bluetooth-Parëzim automatik - Copy&Paste në Ueb faqe - Ngarkim direkt në YouTube	HTC Magic HTC Hero Samsung Galaxy i7500 LG GW620
1.6 (Donut)	09/2009	- Kërkimi si shërbim - Kalimi i shpejtë në mes të fotove dhe video funksioneve - Nga teksti në fjalë - Përkrahje e gjesteve	Acer Liquid A1
2.0 (Eclair)	10/2009	- Përkrahje e këmbimit valutor - Përkrahje e kombinuar e llogarive të email-it - Funksion kërkimi për SMS/MMS - Zmadhimi dixhital dhe flashpërkamerë - Integrimi i HTML 5 në Ueb brouser - Zmadhim i dyfishtë në Ueb brouser - Bookmarks me shikimi - Përkrahje e Bluetooth 2.1	Motorola Milestone
2.1 (Eclair)	01/2010	- Kryesisht përmirësim gabimesh - Prapavijë me animacione - Më tepër Homescreens	Google Nexus One HTC Desire HTC Legend Motorola Defy Samsung I5800 Galaxy Samsung I9000 Galaxy
2.2 (Froyo)	05/2010	- Përkrahje për flash - Aplikimi për backup të të dhënave - Android-Market via PC-Browser, transferimin e drejtpërdrejtë për celular - Musik-Streaming nga PC	Samsung Galaxy Tab HTC Desire HD Motorola Milestone 2 Samsung Galaxy 551 LG P920 Optimus

		- Shfletim më i shpejtë - Apps në kartën SD - WLAN-Hotspot-Feature	3D
2.3 / 2.3.3 / 2.3.4 / 2.3.5 / 2.3.6 (Gingerbread)	12/2010	- VoIP-Client - Menaxhment për optimizim të energjisë - Përkrahje NFC	Google Nexus S HTC Evo 3D HTC Sensation HTC Desire S Motorola Defy+ Samsung Galaxy S II Samsung Galaxy Note Huawei Ideos X3
3.0 (Honeycomb)	Q1/2011	- Optimizuar për tablet	Motorola Xoom HTC Flyer Samsung Galaxy Tab 10.1
3.1 (Honeycomb)	05/2011	- Fine-tuning ndërfaqen e përdoruesit - Përkrahje për USB-Hub - Përkrahje për Google-TV	Sony Tablet S Lenovo ThinkPad Tablet Lenovo IdeaPad K1
3.2 (Honeycomb)	07/2011	- Zmadhim kompatibël: Përmirësim i dukjes së aplikimeve - API të avancuar për ekrane me madhësi të ndryshme - Qasje aplikacioni në SD-Kartë - Përkrahje për tableta të ndryshëm	Sony Tablet P ViewSonic Viewpad 7x Samsung Galaxy Tab 7.7
4.0 (Ice Cream Sandwich)	10/2011	- Dizajn i ri	Google Galaxy Nexus Samsung Galaxy S3
4.1 (Jelly Bean)	06/2012	- Performancë e optimizuar - Mundësia e caktimit të gjuhës offline - Ndryshim kamere - Ndryshim ki Google-Beam-(NFC) - Funksion i zgjeruar kërkimi - Personalizim i informacioneve në Google	Google Nexus 7 Sony Xperia Z HTC One
4.2	11/2012	- Qasje të shpejtë në ndërfaqet	Google Nexus 4

(Jelly Bean)		radio - Ndryshime vizuale te mesazhet - Widgets on Lockscreen - Foto panoramike - Gmail-Client with Pinch-to-Zoom - Përkrahje e më tepër se një llogarie të përdoruesit	Google Nexus 10 Samsung Galaxy S4
4.3 **(Jelly Bean)**	07/2013	- eingeschränkte Benutzerkonten (nur Tablets) - Përshpejtim harduerik i shikimit të DRM-Videove - Përmes T9 kërkimi në kontakte tek aplikacionet e nexus-it	Google Nexus 7 (2013) Samsung Galaxy Note 3 Motorola Moto G
4.4 **(Kitkat)**	11/2013	- Multi-Tasking i shpejtë - Funksion printimi - Ndërrim dizajni	Google Nexus 5
5.0 **(Lollipop)**	10/2014	- Pamje e re: Material Design - Nivel i ri ekzekutimi: Android Runtime (ART) - Project Volta: konsumim energjie i optimizuar - Heads-Up-Notifications	Google Nexus 6 Google Nexus 9 Samsung Galaxy S6 HTC One (M9)
5.1 **(Lollipop)**	03/2015	- Përmirësim gabimesh	LG G4
6.0 **(Marshmallow)**	10/2015	- Menaxhimi i ndryshuar i të drejtave (App Permissions) - Update i aplikacionit për shenjë - Mekanizma të rinj për kursim energjie	Google Nexus 5X Google Nexus 6P Google Pixel C
7.0 **(Nougat)**	08/2016	- Multi-Windows-Modus - Updates në prapavijë - 1500 Smileys të reja - Modus i optimizuar për kursim energjie.	

Tabela 19: Versionet e sistemit operativ Android dhe karakteristikat e tij

(Trautmann, 2016)

Aplikacioni i Android-it përmes të cilit shërbehet përdoruesi, shfrytëzon algoritmin gjenetik si ueb shërbim për të llogaritur turin optimal, por në të njëjtën kohë shfrytëzon edhe ueb shërbimet e Google-it: Google Maps Android, Overlayer (Markers, Polyline dhe Info Window), Directions dhe Google-Places-API për Android.

Këtu është bërë integrimi i ueb shërbimeve të Google-it në aplikacionin e Androidit në mënyrë që t'i ofrojë përdoruesit një ndërfaqe sa më interaktive me aplikacionin, kështu që ai ta ketë të lehtë dhe në mënyrë intuitive përdorimin e tij.

Programimi i Android-Manifestit

Programimi i një aplikacioni të Androidit fillon me programimin Manifestit.
Në Manifest të Android-it ruhen të dhënat të rëndësishme në lidhje me aplikacionin si dhe i lejohet qasja në shërbimet e Google-it. Pra, Android-Manifest është skedar XML-i, i cili fillon me tag-un, "manifest'' dhe mandej pasojnë informacionet në lidhje me paketën e aplikimit dhe versionit të tij. Më vonë duhet të pasojë deklarimi dhe definimi i aktiviteteve të aplikacionit, përndryshe i gjithë aplikimi do të rrëzohet nëse bën përpjekje ta startojë një aktivitet të programuar, por të padeklaruar këtu në Manifest.
Mundësitë të tjera përshtatjeje mund të ndërmerren në Manifest:

- **Themes** – janë shabllone të ndryshme dizajni që mund ta ndryshojnë pamjen e aplikimit si dhe të aktiviteteve të ndryshme individuale.
- **Orientimi i ekranit –** këtu mund të ndalohet rrotullimi i ekranit dhe lë të shfaqur aplikimin vetëm në modalitetin portret.

- **Permissions –** po ashtu edhe lejet duhet të specifikohen në manifestin e Android-it. Pra, çfarë i lejohet aplikimit të bëjë me pajisjen tuaj. P. sh.: të ketë qasje në kontaktet personale, në internet etj.

Manifest - XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="phd.laurikhelshani"
              android:versionCode= "1"
              android:versionName= "1.0" >
<uses-sdk android:minSdkVersion= "8" />
<uses-permission android:name = "android.permission.INTERNET" />
<uses-permission android:name =
 "android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name =
"android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name =
 "android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name =
 "android.permission.ACCESS_FINE_LOCATION"/>

<uses-feature
      android:glEsVersion = "0x00020000"
      android:required = "true" />

<application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

<activity
         android:name=" phd.laurikhelshani.MainActivity"
         android:label="@string/app_name"
         android:launchMode="singleTask" >
<intent-filter>
<action android:name="android.intent.action.MAIN"
              android:launchMode="singleTask"/>
```

```
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity
        android:name=" phd.laurikhelshani.Preferences"
        android:parentActivityName=" phd.laurikhelshani.MainActivity" >
</activity>
<meta-data android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
<meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="MY_API_KEY"/>
</application>
</manifest>
```

Tabela 20: Programimi i Manifestit të Androidit përmes XML-it

Komente në lidhje me skedarin konkret të manifestit të aplikacionit

Emri i paketës: phd.laurikhelshani, ndërsa versioni i kodit është 1.

Përmes komandës: *uses-sdk android: minSdkVersion="8"* deklaroj që vetëm pajisjet me sistem operativ Android 2.3 dhe më lartë janë kompatibël dhe do ta përkrahin këtë aplikim, ndërsa versionet e Androidit me të vogla se 2.3 nuk i përkrahin Google Play Services APIs. Për të zhvilluar një aplikim duke përdorur shërbimet Google Play services APIs, i gjithë projekti duhet të ngrihet bazuar në Google Play services SDK, e cila duhet të jetë e instaluar që më parë në kompjuter.

Më pas pason dhënia e lejeve, se çfarë i lejohet aplikimit të bëjë me pajisjen, përmes komandës: *uses-permission android* dhe duke cekur emrin specifik të lejes. Pra, këtij aplikimi i lejohet të ketë qasje në internet, të verifikojë nëse pajisja është e lidhur në wireless, të shkruajë në memorien interne, të gjejë vendndodhjen aktuale të pajisjes etj.

Në mes të tag-eve “<activity></activity>” janë dy aktivitete të aplikimit. Njëri prej tyre është kryesor ku paraqitet harta e Google-it, përmes të cilit përdoruesi shërbehet, ndërsa tjetri është dytësor ku përdoruesi ja përshtat funksionalitetin vetvetes.

Përmes komandës: *android:launchMode = "singleTask"*, është siguruar që të krijohet vetëm një objekt i aktivitetit parësor (kryesor).

Për të përdorur Google Maps Android API, projekti është regjistruar në Console Google API, meqë kjo është e domosdoshme për marrjen e një çelësi dhe për shtimin e tij në projekt. Ka lloje të ndryshme të kufizimeve për çelësat API, mirëpo në këtë rast është dashur një API_KEY me kufizim për aplikime Android-i. Përmes këtij çelësi Google numëron që sa herë aplikimi ka pasur qasje në hartën e Google-it, meqë numri i qasjeve falas në hartë brenda 24 orëve është i kufizuar. Nëse tejkalohet limiti dhe nuk keni kredi në llogarinë e Google-it, jeni të kufizuar 24 orë të mos keni qasje në të. Si rrjedhojë e saj aplikimi nuk punon, meqë i gjithë funksionaliteti i tij është i mbështetur në shërbimet e Google-it, e posaçërisht në hartat e tij.

```
<meta-data
        android:name = "com.google.android.maps.v2.API_KEY"
        android:value = "MY_API_KEY"/>
```

Programimi i preferencave (parapëlqimeve)

Aplikime të ndryshme shpesh përmbajnë kurdisje (settings) që e lejojnë përdoruesin që të ndryshojë tiparet dhe sjelljet e aplikimit. Edhe në këtë aplikim është programuar një ndërfaqe përmes së cilës përdoruesi mund t'ia përshtasë vetvetes disa funksione të aplikimit.

Sikur çdo ndërfaqe e aplikimeve të tjera, ashtu edhe kjo ndërfaqe e këtij aplikimi kërkon fillimisht dizajnimin e saj dhe mandej përmes programimit edhe funksionalitetin. Dizajnimi i ndërfaqës është bërë prapë përmes XML-it, ndërsa funksionaliteti është programuar në JAVA.

Dizajnimi i parapëlqimeve (preferencave)

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
   xmlns:tools = "http://schemas.android.com/tools"
   android:layout_width = "match_parent"
   android:layout_height = "match_parent"
   android:paddingBottom = "@dimen/activity_vertical_margin"
   android:paddingLeft = "@dimen/activity_horizontal_margin"
   android:paddingRight = "@dimen/activity_horizontal_margin"
   android:paddingTop = "@dimen/activity_vertical_margin"
   tools:context = ".MainActivity" >

<EditText
      android:layout_width = "fill_parent"
      android:layout_height = "50dp"
      android:id = "@+id/url"
      android:singleLine = "true"
      android:inputType = "textCapWords"
      android:hint = "URL-ja e serverit" />

<EditText
      android:layout_width = "fill_parent"
      android:layout_height = "50dp"
      android:layout_below = "@+id/url"
      android:hint = "Porti i serverit (Numrat prej 1-10)"
      android:id = "@+id/port"
```

```
        android:digits = "0123456789"
        android:inputType = "phone" />

<EditText
        android:layout_width = "fill_parent"
        android:layout_height = "50dp"
        android:layout_below = "@+id/port"
        android:id = "@+id/celesi"
        android:hint = "Celesi per qasje ne harte Google-i" />

<CheckBox
        android:id = "@+id/erresim"
        android:layout_width = "fill_parent"
        android:layout_height = "50dp"
        android:layout_below = "@+id/celesi"
        android:text = "Te mos erresohet ekrani pas nje kohe te mosaktivitetit me
aplikacionin" />


<Button
        android:layout_width = "fill_parent"
        android:layout_height = "50dp"
        android:layout_below = "@+id/erresim"
        android:id = "@+id/ruajBtn"
        android:text = "Ruaj" />

<Button
        android:layout_width = "fill_parent"
        android:layout_height = "50dp"
        android:layout_below = "@+id/ruajBtn"
        android:id = "@+id/pastroBtn"
        android:text = "Pastro" />

<Button
        android:layout_width = "fill_parent"
        android:layout_height = "50dp"
        android:layout_below = "@+id/pastroBtn"
        android:id = "@+id/prapaBtn"
        android:text = "Prapa" />

</RelativeLayout>
```

Tabela 21: Programimi i ndërfaqes së preferencave përmes XML-it

Siç mund të dallohet lehtë nga skedari i mësipërm i XML-it, janë dizajnuar disa fusha dhe pulla. Fusha e parë shërben për të shkruar IP – adresën ose URL, ku ofrohet algoritmi gjenetik si ueb shërbim. Në këtë fushë mund të shkruhen shkronja dhe numra. Nëse klikohet në këtë fushë për të shkruar, shfaqet tastiera e Androidit me shkronja dhe numra.

Fusha e dytë është paraparë për të dhënë numrin e portit të serverit ku ofrohet algoritmi gjenetik si ueb shërbim. Këtu mund të shkruhen vetëm numra nga 0 deri në 9. Me të klikuar në këtë fushë për të shkruar, shfaqet tastiera e Androidit vetëm me numra. Çdo herë edhe pas fshirjes së preferencave, këtu shfaqet porti default 8080.

Parapëlqimi apo preferenca e tretë është programuar që përmes një check-boxi përdoruesi mund të vendosë nëse duhet t'i errësohet ekrani i pajisjes pas një kohe të mosaktivitetit me aplikimin, për të kursyer kështu baterinë e pajisjes.

Parapëlqimi i fundit është fusha ku përdoruesi mund të shkruajë ndonjë çelës të vetin përmes të cilit aplikimit i lejohet qasja në harta të Google-it. Nëse këtu nuk vendoset ndonjë çelës, atëherë aplikimi punon me një çelës i cili është para programuar.

Pulla ,"Ruaj'' shërben për të ruajtur vlerat e vendosura të preferencave që të zbatohen në aplikim dhe bën fshehjen e tastierës së paraqitur për të shënuar.

Pulla ,"Fshij'' shërben për fshirjen e vlerave të vendosura të preferencave, ndërsa pulla , "Prapa'' shërben për t'u kthyer prapa tek aktiviteti kryesor. Që të dyja pas kryerjes së funksionit të tyre, bëjnë edhe heqjen nga ekrani të tastierës, nëse ajo është shfaqur për të shënuar apo ndryshuar vlerat e preferencave.

Këto preferenca janë programuar që aplikimi i Android-it të bëhet sa më i përgjithshëm dhe për të mos pasur nevojë që të riprogramohet, rikompilohet dhe të instalohet prapë në pajisje

sa herë që ndërron IP-ja, hosti apo porti i serverit ku algoritmi gjenetik ofrohet si ueb shërbim.

Shkëputje nga kodi i JAVA-së së preferencave

```
protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.preferences);
		context = this.getApplicationContext();
		url = (EditText) findViewById(R.id.url);
		porti = (EditText) findViewById(R.id.port);
		celesi = (EditText) findViewById(R.id.celesi);
		erresim = (CheckBox) findViewById(R.id.erresim);
		pullaRuaj = (Button) findViewById(R.id.ruajBtn);
		pullaPastro = (Button) findViewById(R.id.pastroBtn);
		pullaPrapa = (Button) findViewById(R.id.prapaBtn);

		sharedPref = getSharedPreferences("parapelqimet", MODE_PRIVATE);

		erresim.setChecked(sharedPref.getBoolean("erresim", false));
		url.setText(sharedPref.getString("url", ""));
		porti.setText(Integer.toString(sharedPref.getInt("porti", 0)));
	String key_map = sharedPref.getString("celesi", null);

		pullaPrapa.setOnClickListener(new OnClickListener() {
			public void onClick(View v) {
				Intent myIntent = new Intent(context, MainActivity.class);
				startActivity(myIntent);
			}
		});

		pullaPastro.setOnClickListener(new OnClickListener() {
			public void onClick(View v) {
				url.setText("");
				porti.setText("");
				celesi.setText("");
				erresim.setChecked(false);
				SharedPreferences.Editor editor = sharedPref.edit();
				editor.clear();
				editor.commit();
				InputMethodManager imm =
(InputMethodManager)getSystemService(Context.INPUT_METHOD_SERVICE);
				imm.hideSoftInputFromWindow(v.getWindowToken(), 0);
				Toast msg = Toast.makeText(getBaseContext(), "Shenimet
```

```
u pastruan me sukses.", Toast.LENGTH_LONG);
                                msg.setGravity(Gravity.CENTER, 0, 0);

                                msg.show();
                        }
                });

                pullaRuaj.setOnClickListener(new OnClickListener() {
                        public void onClick(View v) {
                                SharedPreferences.Editor editor = sharedPref.edit();
                                editor.putString("url", url.getText().toString());
                                editor.putString("celesi", celesi.getText().toString());
                                editor.putInt("porti",
Integer.parseInt(porti.getText().toString()));
                                editor.putBoolean("erresim", erresim.isChecked());
                                editor.commit();
                                InputMethodManager imm =
(InputMethodManager)getSystemService(Context.INPUT_METHOD_SERVICE);
                              imm.hideSoftInputFromWindow(v.getWindowToken(), 0);
                                Toast msg = Toast.makeText(getBaseContext(), "Shenimet
u ruajten me sukses.", Toast.LENGTH_LONG);
                                msg.setGravity(Gravity.CENTER, 0, 0);
                                msg.show();
                        }
                });
        }
```

Tabela 22: Programimi i funksionalitetit të preferencave me JAVA

Figura 17: GUI i preferencave

Ky aplikacion është klient i dyfishtë, së pari në raport me shfrytëzimin e ueb shërbimeve të Google-it si dhe në raport me konsumimin e algoritmit gjenetik si ueb shërbim për optimizimin e rrugës. Pra,është aplikacioni më kompleks që kam programuar për pjesën praktike të doktoratës sime.

Në këtë aplikacion përmes programimit janë integruar hartat dhe ueb shërbimet e Google-it për ta bërë aplikacionin sa më interaktiv dhe më intuitiv për përdoruesit e tij. Pra, përmes këtij aplikacioni përdoruesit (shëtitësit) i mundësohet që në mënyre intuitive të ketë mundësinë e zgjedhjes së vendeve që dëshiron të vizitojë dhe kështu ta lerë aplikacionin që t'ia optimizojë këtë rrugë, dhe në mënyrë grafike, në formë harte, t'i vizatohet rruga që duhet ndjekur në mënyrë që ajo të jetë më e shkurtra e mundshme.

Përdoruesi (shëtitësi) ka dy mundësi zgjedhjeje të vendeve që dëshiron t'i vizitojë.

1) Ai e di saktësisht në hartë vendndodhjen e vendeve që dëshiron t'i vizitojë dhe përmes prekjes me gisht mbi hartën e Google-it, ai shënjon vendet që dëshiron t'i vizitojë. Në këtë rast duhet të nxirren koordinatat gjeografike të vendit të zgjedhur nga harta e integruar e Google-it në aplikacionin e androidit dhe duke përdorur event-in onMapLongClick, ueb shërbimin e Google-it Overlayer (Marker), bëhet shënjimi i atij vendi.

Programimi i onMapLongClick

```
@Override
public void onMapLongClick(LatLng point) {
    try  {
        GoogleMap myMap;
        String key = "MyApiKey";
        String url_geocoding = "https://maps.googleapis.com/maps/api/geocode/json";
        StringBuilder builder = new StringBuilder();
        builder.append(url_geocoding).append("?latlng=")
        .append(point.latitude).append(",").append(point.longitude);
        builder.append("&sensor=true");
        builder.append("&key=").append(key);
        JSONObject objekti = read(builder.toString());
        JSONArray arr = objekti.getJSONArray("results");
        String address = arr.getJSONObject(0).getString("formatted_address");
        JSONObject components = arr.getJSONObject(1);
        JSONArray address_components=
            components.getJSONArray("address_components");
        String place_name =
            address_components.getJSONObject().getString("long_name");
        myMap.addMarker(new
      MarkerOptions().position(point).title(place_name).snippet("Address: "
+address));
    } catch (Exception e) {
       Log.e("Error","Response string buffer error. "+ e.getMessage());
    }
```

```
}
```

Tabela 23: Programimi i event-it onMapLongClick

2) Përdoruesi nuk e di se ku ndodhen në hartë vendet që dëshiron t'i vizitojë, por ai di vetëm adresën, pjesë adrese, emrin e vendit që dëshiron ta vizitojë, qoftë ajo bukuri natyrore, vend me vlerë historike etj. Pra, përmes funksionit për kërkim duke përdorur ueb shërbimin e Google-it: Google Places API, i cili lejon që të kërkojmë në bazën e të dhënave të Google-it në bazë të mostrës për kërkim, i propozohen përdoruesit vendet e gjetura. Pasi ai të zgjedhë, nga lista e vendeve të propozuara, vendin e duhur, atëherë duhet që ai vend të shënjohet në hartë të Google-it dhe pamja të zhvendoset tek ai vend.

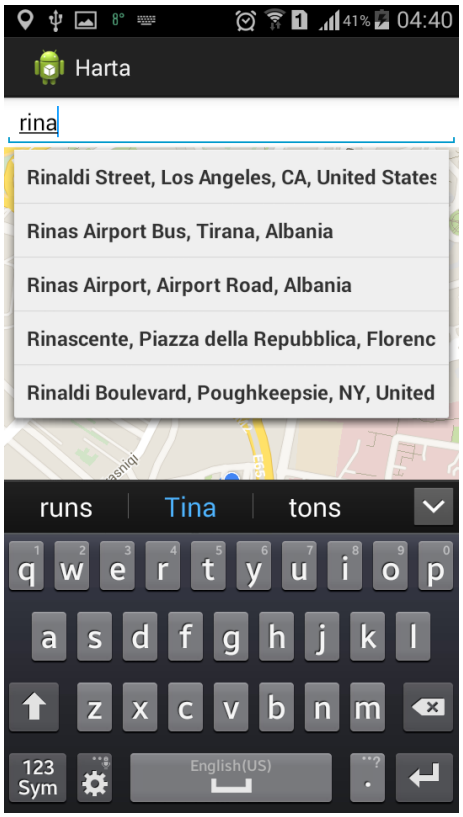
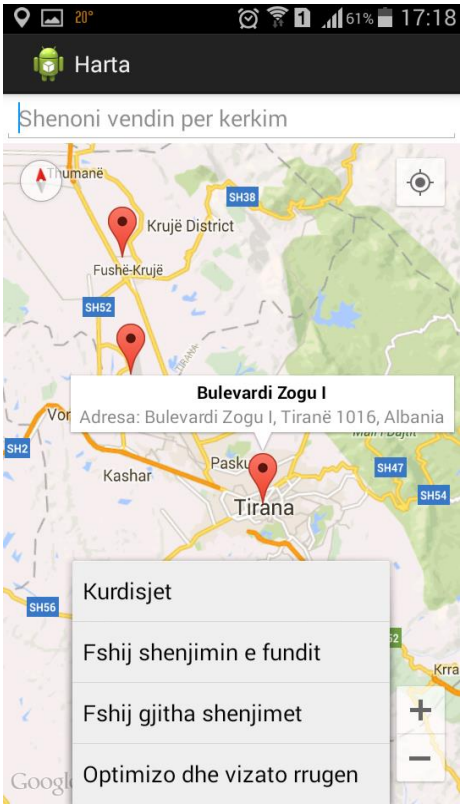


Figura 18: Pamje nga aplikacioni i androidit gjatë përzgjedhjes së endeve për vizitë përmes kërkimit



Pasi përdoruesi t'i ketë zgjedhur vendet që dëshiron të vizitojë, në meny zgjedh funksionin: 'Optimizo dhe vizato rrugën', aplikacioni mbledh të gjitha informatat e vendeve për t'u vizituar nga harta e Google-it, i konverton ato në formatin JSON dhe përmes kërkesës HTTP-Post konsumon ueb shërbimin e ofruar si RESTful (algoritmin gjenetik).

HTTP- POST

```
public static String POST(String url, JSONObject objekti) {
   InputStream inputStream = null;
   String result = "";
      try {
         HttpClient httpclient = new DefaultHttpClient();
         HttpPost httpPost = new HttpPost(url);
         String json = "";
         json = objekti.toString();
         StringEntity se = new StringEntity(json);
         httpPost.setEntity(se);
         httpPost.setHeader("Accept", "application/json");
         httpPost.setHeader("Content-type", "application/json");
         HttpResponse httpResponse = httpclient.execute(httpPost);
         inputStream = httpResponse.getEntity().getContent();
         if(inputStream != null)
              result = convertInputStreamToString(inputStream);
         else
              result = "Nuk punon!";
      } catch (Exception e) {
         Log.d("InputStream", e.getLocalizedMessage());
      }
   return result;
}
```

Tabela 24: Funksioni HTTP - POST (JAVA)

Përgjigjja e kthyer nga ueb shërbimi përpunohet dhe përshtatet në mënyrë që të përpunohet më tutje. Për çdo dy palë koordinata aplikacioni shfrytëzon ueb shërbimin e Google-it: Directions, që të gjejë rrugën nga pika A në pikën B dhe përmes Polylines (seri vizash të drejta) vizaton rrugën që duhet ndjekur në hartën e Google-it.

Rruga e optimizuar mes kryeqyteteve shqiptare është paraqitur më poshtë në aplikacionin e androidit, figurë e cila në mënyrë grafike sqaron punën e aplikacionit dhe vizatimin e rrugës së optimizuar.

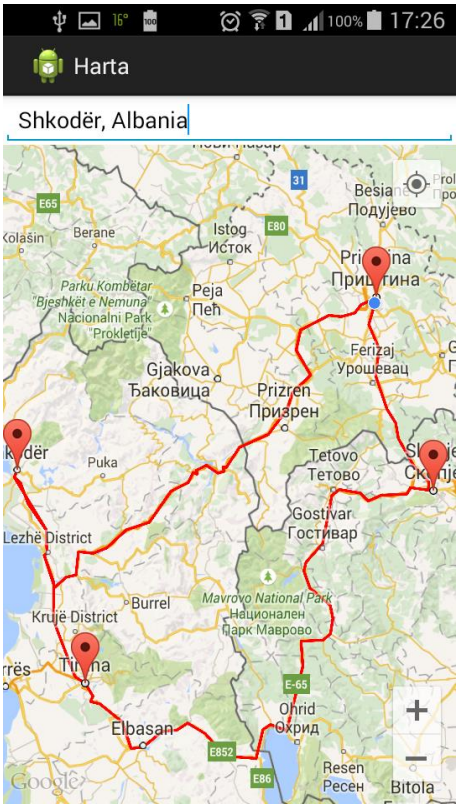


Figura 19: Rruga e optimizuar në aplikacionin klient të Androidit

4.3. Përmbledhje

Sistemi është programuar duke u bazuar në një arkitekturë tashmë të mirënjohur klient-server, ku algoritmi gjenetik është programuar në gjuhën programuese JAVA dhe i tëri ofrohet si ueb shërbim RESTful në serverin TOMCAT përmes NetBeans-IDE.

Në fillim u sqarua ndërlidhja në mes problemit biznesmeni udhëtar dhe teorisë së grafeve si parakusht për ta aplikuar algoritmin gjenetik mbi grafe.

Meqë përmes algoritmit gjenetik programuesi mundohet të simulojë një pjesë të natyrës në kompjuter (evolucionin- ku më i forti mbijeton), atëherë programimi i këtij algoritmi duhet të simulojë këto faza: gjenerimin e popullatës, përzgjedhjen në mënyrë të rastësishme, bashkëdyzimin, ndryshimin e rastësishëm në gjene, vlerësimin si dhe hedhjen apo zëvendësimin e individit të ri mes të dy individëve të përzgjedhur më parë.

U diskutua po ashtu në lidhje me parametrat që ndikojnë në punën e algoritmit gjenetik e që janë:madhësia e popullatës, fqinjësia/madhësia e grupit, përqindja e mutacionit, numri i vendeve fqinje dhe numri maksimal i brezave.

Fillimisht u sqarua programimi i klasës 'Vendi', si klasë bazë për ruajtjen e të gjitha shënimeve të një vendi të Google-it, së bashku me funksionet përkatëse dhe parametrat e tyre hyrës.

Klasa 'Tur' është strukturë që përmban informacionet në lidhje me turin dhe lejon manipulimin me të përmes funksioneve përkatëse. Klasa 'Popullata' është struktura ku ruhen disa zgjidhje të mundshme për t'u manipuluar me to, dhe në këtë mënyrë konvergjohet drejt zgjidhjes optimale. Klasa 'AlgoritmiGjenetik' evoluon me popullatën. Ndërlidhja e këtyre klasave dhe bashkëveprimi në mes tyre u sqarua përmes UML – diagramit të klasave.

Në këtë kapitull u sqarua po ashtu programimi i algoritmit gjenetik në dy versione. Që të dyja versionet funksionet bazë i kanë të njëjta, mirëpo dallojnë te programimi i funksionit për vlerësim (fitness). Secili nga ata ka përparësitë dhe mangësitë e veta.

Versioni, i cili përdor formulën e Haversinës për llogaritjen e gjatësisë së turit, ka përparësinë e vet, që e bën algoritmin gjenetik të punojë më shpejt, mirëpo nuk merr parasysh rrugët reale në terren dhe si rrjedhojë marrim si rezultat një gjatësi joreale të turit.

Versioni tjetër, i cili te funksioni për llogaritjen e gjatësisë së turit, shfrytëzon ueb shërbimin e Google-it: Google Distance Matrix kthen rezultat ekzakt, mirëpo bën që puna e algoritmit gjenetik të ngadalësohet, por dhe të jetë i zbatueshëm vetëm për vendet anëtare të OKB-së, meqë ky shërbim ofrohet vetëm për vende të tilla.

Përmes shkëputjes së një pjese të kodit u sqarua se si i gjithë algoritmi u ofrua si ueb shërbim RESTful përmes JAX-RS.

Androidi është sistem operativ me kod burimor të hapur dhe si bazë ka një sistem tjetër operativ të quajtur Linux. Zhvillimi i këtij sistemi operativ (Androidit) është iniciuar dhe përkrahet nga Google.

U sqarua po ashtu se çdo zhvillues apo programues aplikacionesh të Androidit programimin duhet ta fillojë nga skedari i Manifestit të Android-it, skedar ky ku ruhen të dhënat e rëndësishme në lidhje me aplikacionin, të cilit i lejohet qasja në shërbimet e Google-it. Në këtë skedar XML-i duhet të definohen aktivitetet e aplikacionit si dhe sendet plotësuese: themes, orientimi i ekranit si dhe premissions.

Përmes zhvillimit të këtij aplikacioni është tentuar që kjo doktoraturë të mos mbetet vetëm kontribut intelektual, por edhe të jetësohet. Pra, ky aplikacion Androidi, përmes të cilit shërbehet përdoruesi, shfrytëzon algoritmin si ueb shërbim për të llogaritur turin optimal, por dhe shfrytëzon në të njëjtën kohë edhe ueb shërbimet e Google-it: Google Maps Android, Overlayer (Markers, Polyline dhe Info Window), Directions dhe Google-Places-API për Android.

Në këtë aplikacion Androidi janë integruar ueb shërbimet e Google-it në mënyrë që përdoruesit t'i ofrohet një ndërfaqe sa më interaktive me aplikacionin, kështu që ai ta këtë më të lehtë dhe në mënyrë intuitive përdorimin e tij.

Në këtë kapitull përmes shkëputjes së pjesëve të kodit burimor u sqarua si programohet një event në hartën e Google-it, manifesti dhe preferencat, ndërsa përmes screenshots sqarohet sesi përdoruesi mund të shërbehet me të.

Në fillim u sqarua ndërlidhja në mes të problemit biznesmeni udhëtar dhe teorisë së grafeve si parakusht për ta aplikuar algoritmin gjenetik mbi grafe.

Meqë përmes algoritmit gjenetik programuesi mundohet të simulojë një pjesë të natyrës në kompjuter (evolucionin- ku më i forti mbijeton), atëherë programimi i këtij algoritmi duhet të simulojë këto faza: gjenerimin e popullatës, përzgjedhjen në mënyrë të rastësishme, bashkëdyzimin, ndryshimin e rastësishëm në gjene, vlerësimin si dhe hedhjen apo zëvendësimin e individ të ri me të dy individët e përzgjedhur më parë.

U diskutua po ashtu në lidhje me parametrat që ndikojnë në punën e algoritmit gjenetik e që janë:madhësia e populatës, fqinjësia/madhësia e grupit, përqindja e mutacionit, numri i vendeve fqinje dhe numri maksimal i brezave.

Fillimisht u sqarua programimi i klasës 'Vendi', si klasë bazë për ruajtjen e të gjitha shënimeve të një vendi të Google-it, së bashku me funksionet përkatëse dhe parametrat e tyre hyrës.

Klasa 'TIR' është strukturë që përmban informacionet në lidhje me turin dhe lejon manipulimin me të përmes funksioneve përkatëse. Klasa 'Popullata' është struktura ku ruhen disa zgjidhje të mundshme për t'u manipuluar me to, dhe në këtë mënyrë konvergjohet drejt zgjidhjes optimale. Klasa 'AlgoritmiGjenetik' evoluon me popullatën. Ndërlidhja e këtyre klasave dhe bashkëveprimi në mes tyre u sqarua përmes UML – diagramit të klasave.

Në këtë kapitull u sqarua po ashtu programimi i algoritmit gjenetik në dy versione. Që të dyja versionet funksionet bazë i kanë të njëjta, mirëpo dallojnë te programimi i funksionit për vlerësim (fitness). Secili nga ta ka përparësitë dhe mangësitë e veta.
Versioni i cili përdorë formulën e Haversinës për llogaritjen e gjatësisë së turit, ka përparësinë e vet që bën algoritmin gjenetik të punojë më shpejtë, mirëpo nuk merr parasysh rrugët reale në terren dhe si rrjedhojë marrim si rezultat një gjatësi jo reale të turit.
Versioni tjetër i cili te funksioni për llogaritjen e gjatësisë së turit, shfrytëzon ueb shërbimin e Google-it: Google Distance Matrix kthen rezultat ekzakt, mirëpo bën që puna e algoritmit

gjenetik të ngadalësohet si dhe të jetë i zbatueshëm vetëm për vendet anëtare të OKB-s, meqë ky shërbim ofrohet vetëm për vendet e tilla.

Së fundmi përmes shkëputjes së një pjese të kodit u sqarua se si i gjithë algoritmi u ofrua si ueb shërbim RESTful përmes JA-KS.

Në kapitullin vijues do të lexoni në lidhje me aplikacionin e androidit përmes të cilit shërbehet përdoruesi që shfrytëzon algoritmin si ueb shërbim për të llogaritur turin optimal dhe në të njëjtën kohë shfrytëzon edhe ueb shërbimet e Google-it: Google Maps Android, Overlayer (Markers, Polyline dhe Info Window), Directions dhe Google-Places-API për Android.

KAPITULLI V: ANALIZË, TESTIM DHE INTERPRETIM REZULTATESH

Simulatorin për testimin e algoritmit gjenetik e kam programuar me gjuhën programuese JAVA dhe duke përdorur librarin 2D të po së njëjtës gjuhë, për të matur optimizimin në përqindje si dhe për të demonstruar grafikisht punën e algoritmit duke qenë ai në ekzekutim e sipër.

Meqë në shumicën e punimeve të ngjashme shkencore është arritur në përfundimin se algoritmi gjenetik është i zbatueshëm për zgjidhjen e problemit biznesmeni udhëtar, mirëpo në asnjërën prej tyre nuk tregohet përqindja e optimizimit si dhe koha e ekzekutimit.

Përmes këtij simulatori kjo tezë doktorature është vërtetuar vetëm në mënyrë praktike sepse ende nuk ka filluar puna praktike që të jetësohet dhe mos të mos mbetet vetëm si punim shkencor.

Më pas është testuar me po të njëjtën metodë një algoritëm tjetër alternativ dhe kam treguar po ashtu që edhe ai është i zbatueshëm në zgjidhjen e problemit TSP. Algoritmi quhet Simulated Annealing i cili po ashtu jep zgjidhje të përafërta dhe joekzakte si ai gjenetik.

Gjetja e një algoritmi tjetër për optimizim dhe krahasimi i tij me algoritmin gjenetik ishte njëra nga objektivat e fundit të punës praktike të doktoratës sime.

Duhet pasur parasysh se rezultatet e mëposhtme në formë tabelore i kam përftuar duke ekzekutuar simulatorin që kam programuar për testim në kompjuter laptop me karakteristikat e mëposhtme:

Sistemi operativ: Windows 7 Ultimate, 32-bit

CPU: Intel Core 2 Duo, 183 GHz

RAM: 4 GB

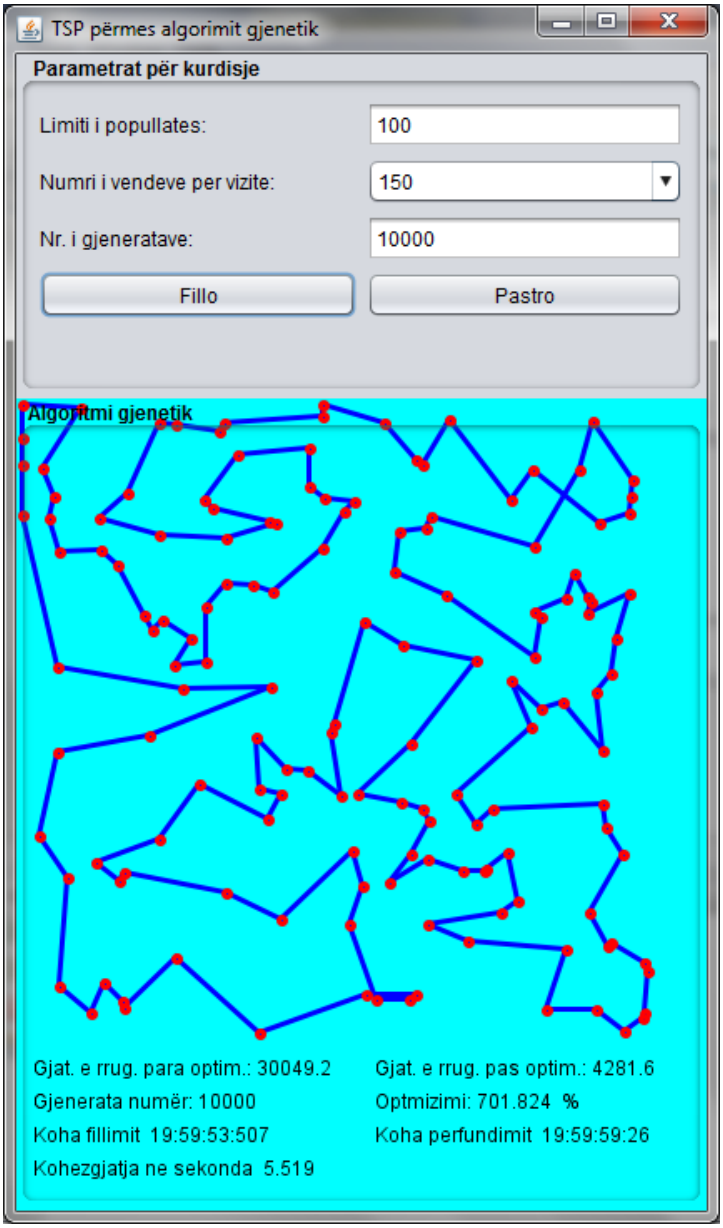


Figura 20: Simulatori për testimin e algoritmit gjenetik

# vendeve	Madhësia popullatës	# gjeneratave	Gjatësia e rrugës fillestare	Gjat. e rrugës pas optimizi-mit	Optimizimi në %	Kohëzgjatja e ekzekutimit në sekonda
30	1000	1000	6785.0	1854.4	365.878	1.842
50	1000	1000	9580.9	2135.7	448.611	2.732
70	1000	1000	13702.4	2611.1	524.783	2.589
100	1000	1000	19606.8	3251.9	602.932	4.558
120	1000	1000	22633.4	3790.6	597.097	4.997
150	1000	1000	31768.2	5357.5	592.973	4.900
100	1000	5000	20520.3	2982.3	688.068	23.201
120	1000	6000	24327.8	3689.2	659.436	22.930
150	1000	10000	29131.2	3909.0	745.230	57.902
150	50	10000	20520.3	2982.3	658.004	3.293
150	100	10000	30758.7	4120.4	746.499	57.902
150	150	10000	29703.1	4251.3	698.681	7.420

Tabela 25: Rastet për testim dhe rezultatet e algoritmit gjenetik

5.1. Interpretim rezultatesh të algoritmit gjenetik

Siç shihet nga tabela e mësipërme tri janë ndryshoret kryesore të cilat ndikojnë drejtpërdrejt në ecurinë e punës së algoritmit, përfundimin e tij si dhe në rezultatin përfundimtar (rrugën e optimizuar të biznesmenit udhëtar). Këto variabla janë: Numri i vendeve për t'u vizituar, madhësia e popullatës, e cila përfaqëson zgjidhjet e mundshme, të cilat algoritmi i përpunon (turet e mundshme të biznesmenit udhëtar) dhe numri i gjeneratave (pas sa orvatjeve për optimizimin e rrugës nga turet e mundshme algoritmi pushon së ekzekutuari).

Programi simulon punën e algoritmit grafikisht duke bërë gjenerimin e rastësishëm të pikave në rrafsh që simbolizojnë vendet për t'u vizituar, dhe tërheq drejtëza në mes të pikave që simbolizojnë rrugët që çojnë te këto vende. Pra, largësia në mes të dy pikave nuk dihet që nga fillimi. Meqë kemi të bëjmë me rrafshin, atëherë çdo pikë adresohet përmes koordinatave X dhe Y. Për të vlerësuar distancën në mes të dy pikave përdor formulën e Euklidit. Pra, kolona "Gjatësia fillestare e rrugës" në tabelën e mësipërme nuk është asgjë tjetër, përveçse shuma e të gjitha distancave të Euklidit. Kjo vlerë llogaritet nga programi në momentin kur ta zgjedhim në GUI numrin e vendeve që dëshirojmë të vizitojmë dhe para se algoritmi gjenetik të fillojë punën e tij. Thënë shkurt, kjo vlerë shfaqet me shfaqjen e grafit fillestar në GUI për simulim dhe njëkohësisht na shërben si vlerë referuese për krahasim me vlerën e gjatësisë së rrugës pas optimizimit përmes algoritmit gjenetik, për të shprehur kështu në përqindje rrugën e optimizuar. Kolona "Gjatësia e rrugës pas optimizimit" tregon gjatësinë e rrugës pas optimizimit përmes algoritmit. Kolona e përqindjes shpreh në përqindje efektin e optimizimit të rrugës përmes algoritmit gjenetik gjatë zgjidhjes së problemit biznesmeni udhëtar. Pra, shpreh në përqindje raportin në mes të dy kolonave të mëparshme.

Siç mund të lexohet nga rreshtat e tabelës, shohim se me rritjen e numrit të vendeve për vizitë dhe mbajtjen konstante të vlerave të dy variablave të tjera, përfitohet një rezultat optimal deri në njëfarë mase, dhe mandej ky optimizim fillon e zvogëlohet me rritjen e mëtejshme të vlerës së variablës "Nr. i vendeve për t'u vizituar".

Nga të dhënat dhe rezultatet e paraqitura në formë tabelore në tabelën e mësipërme, shohim se optimizimi rritet, me rritjen paralele të vlerave të të dyja variablave "nr. e vendeve për

t'u vizituar" dhe "nr. e gjeneratave" dhe mbajtjen konstante të variablës "madhësia e popullatës".

Optimizimi arrin kulmin nëse "nr. e vendeve për t'u vizituar " e mbajmë konstant, dhe rrisim "numrin e madhësisë së popullsisë" për aq sa të jetë i përafërt me numrin e vendeve për t'u vizituar. Po ashtu numri i gjeneratave mbahet konstant, mirëpo duhet të jetë shumë i madh, afër 100000, për ta parandaluar kështu konvergjimin e parakohshëm të algoritmit dhe përfundimin e punës së tij para se ta optmizojë në maksimum rezultatin.

Siç vërehet, po ashtu, nga rezultat tabelore, kompleksiteti i punës së algoritmit rritet dhe së bashku me të edhe kohëzgjatja e ekzekutimit (kolona në sekonda), nëse rrisim madhësinë e popullatës në mënyre që të përfitojmë rezultat sa më të kënaqshëm që i afrohet më tepër rezultatit ekzakt.

5.2. Interpretim rezultatesh të algoritmit Simulated Annealing

# vendeve	Norma e ftohjes	Gjatësia e rrugës fillestare	Gjat. e rrugës pas optimizimit	Optimizimi në %	Kohëzgjatja e ekzekutimit në sekonda
30	$1*e^{-4}$	14.9	4.3	345.360	0.04
30	$1*e^{-5}$	15.2	4.7	323.509	0.335
50	$1*e^{-5}$	26.1	6.4	405.615	0.441
50	$1*e^{-6}$	28.1	6.1	459.755	2.974
100	$1*e^{-4}$	52.5	9.9	530.889	0.167
100	$1*e^{-5}$	56.3	8.2	690.034	0.656
150	$1*e^{-4}$	83.9	15.4	543.696	0.069
150	$1*e^{-5}$	82.2	11.0	747.626	0.915

150	$1*e^{-6}$	82.6	10.2	808.325	6.635
150	$1*e^{-7}$	80.1	10.5	763.005	68.051

Tabela 26: Rastet për testim dhe rezultatet e algoritmit Simulated Annealing

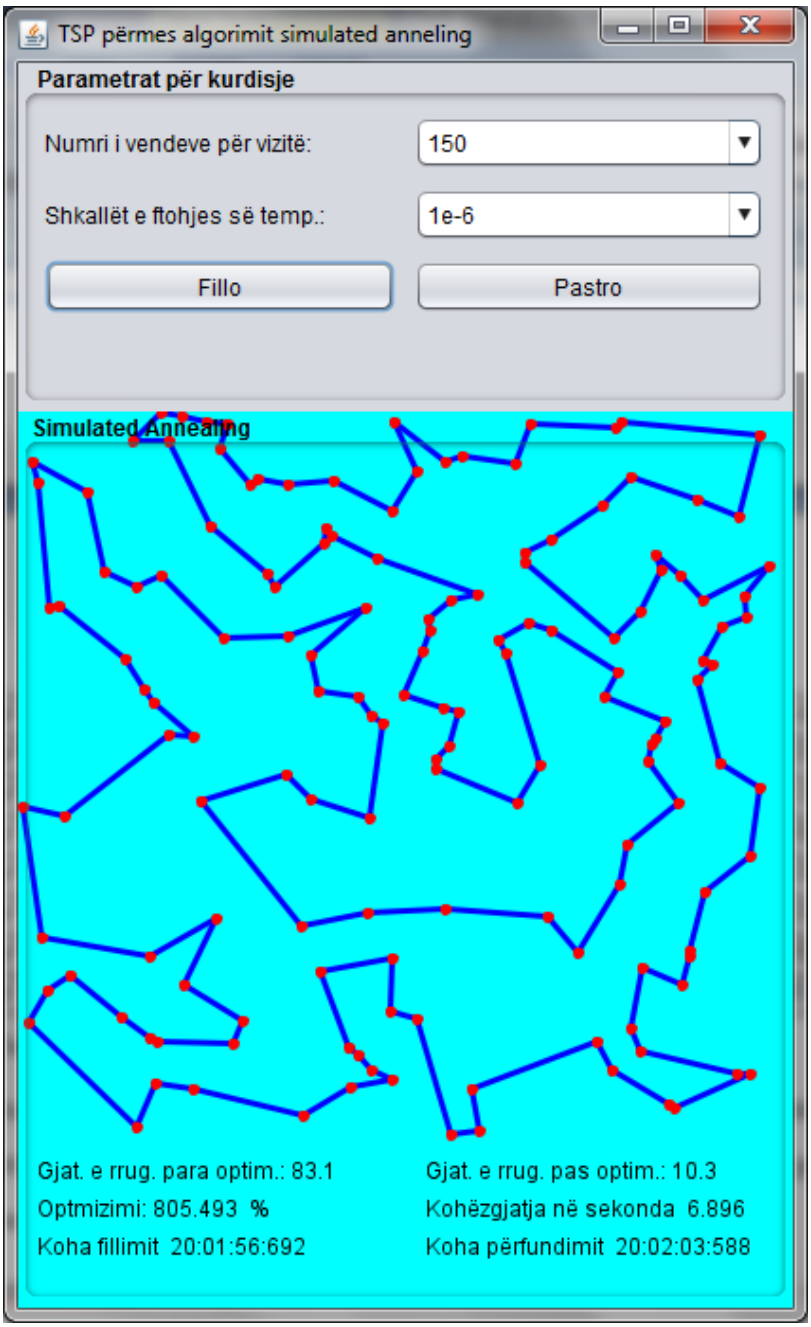


Figura 21: Simulatori për testimin e algoritmit Simulated Annealing

Po e njëjta metodë dhe i njëjti shpjegim vlen edhe për këtë algoritëm si për algoritmin gjenetik, mirëpo bazuar në rezultatet tabelore domosdo duhet të interpretohen disa karakteristika.

Kulminacioni i optimizimit të tij arrin deri në 808.325 % në krahasim me atë të algoritmit gjenetik dhe që është 746.499 %.

Koha e ekzekutimit të algoritmit Simulated Annealing është më e shkurtër sesa koha e atij gjenetik. Pra, për të njëjtin numër të vendeve për t'u vizituar i duhet kohë më e shkurtër për optimizim. Pra, jo më gjatë se 6.635 sekonda.

Kështu optimizimi arrin kulmin në normën e ftohjes $1*e^{-6.}$

Algoritmi Simulated Annealing është shumë më i shpejtë në optimizim. Kështu që, ky algoritëm është disa herë më i shpejtë se ai gjenetik, bazuar në kohën e ekzekutimit.

5.3. Testimi i të gjithë sistemit (punës praktike të doktoratës)

Karakteristikat e mjedisit për testim

Në sistemin që kam programuar, ana e klientit është instaluar në pajisjen me karakteristikat e mëposhtme:

Tablet Samsung Galaxy S3 Lite

Sistemi operativ: Android 4.4.4

Memory: 1GB

CPU: Dual-core 1.2 GHz

Po kështu në sistem, ana e serverit është i instaluar në laptop me këto karakteristika:

Laptop: Dell inspiron 1525

Sistemi operativ: Windows 7 Ultimate, 32-bit

CPU: Intel Core 2 Duo, 1.83 GHz

RAM: 4 GB

Softueri si server: Apache Tomcate 8

Karakteristikat në lidhje me shpejtësinë e rrjetit:

Download: 9.76 Mbit/s

Upload: 1 MBit/s

Shpejtësia mesatare e PING-ut deri te Google Maps: 45 ms

Ecuria e testimit të të gjithë sistemit (cikli komplet)

1) Përdoruesi zgjedh vendet që dëshiron të vizitojë qoftë duke prekur në hartë të Google-it vendet që dëshiron t'i vizitojë, nëse i di saktësisht se ku gjenden ato në hartë, ose përmes funksionit të kërkimit.

2) Pasi t'i ketë zgjedhur të gjitha vendet e dëshiruara për t'i vizituar, në menynë përkatëse përdoruesi zgjedh funksionin optimizo dhe vizato rrugën, me ç'rast fillon edhe matja e kohëzgjatjes së komplet ciklit (gjithë sistemit).
3) Mbledhja e informative nga harta e Google-it, paketimi i tyre në formatin JSON.
4) Dërgimi i të dhënave nga pajisja deri te server-i përmes Wireless-it.
5) Kohëzgjatja e punës së algoritmit gjenetik duke filluar nga konvertimi i të dhënave prej JSON në ArrayList, manipulimi me shënimet e vendeve të Google-it, optimizimi i rrugës dhe paketimi i tyre prapë në JSON.
6) Kohëzgjatja e dërgimit të të dhënave nga server-i te pajisja (tablet)
7) Konvertimi i tyre nga JSON në ArrayList
8) Konsumimi i ueb shërbimit të Google-it DIRECTIONS për çdo dy vende, për ta marrë rrugën ekzakte nga pika A deri te pika B, e për ta vizatuar më pastaj përmes polylines në hartën e Google-it. Pra, kohëzgjatja e këtij procesi ose përmbyllja e këtij cikli.

Pra, kohëzgjatja e ekzekutimit të të gjithë sistemit është shuma e kohëve të lartpërmendura duke filluar prej mbledhjes së të dhënave nga harta e deri te vizatimi i rrugës më të shkurtër (që duhet ndjekur vizitori) në hartën e Google-it.

Pra, kjo le të shërbejë si një raport i një testimi të mirëfilltë duke ia shoqëruar këtij të dhënat për testim, kohëzgjatjet e hapave nga 1 deri në 8, rezultatin e kthyer nga server-i si dhe rrugën e vizatuar në hartë.

Domethënia e shkurtesave që do t'i përdorë në tabelat e mëposhtme:

Koha e **f**illimit të komplet programit = KF

Koha e arritjes së shënimeve në server që shënon njëkohësisht **k**ohën e **f**illimit të punës së **a**lgoritmit gjenetik = KFA

Koha e **p**ërfundimit të punës së **a**lgoritmit gjenetik = KPA

Kohëzgjatja në sekonda apo milisekonda e punës së **a**lgoritmit **gj**enetik, që paraqet diferencën mes KFA-së dhe KPA -së= KAGJ

Koha kur rezultati **a**rrin nga **s**erver-i në **p**ajisje pasi të jetë përpunuar nga algoritmi, e cila njëkohësisht shënon edhe kohën e fillimit të vizatimit të rrugës në hartë, ku përfshihet edhe komunikimi i pajisjes me Server të Google-it për çdo dy vende = KASP

Koha në sekonda nga KF deri në KASP, pra diferenca mes KF-së dhe KASP-së = D

Koha e **p**ërfundimit të **k**omplet **p**rocesit = KPKP

Nga kjo mund të nxirret **k**oha e nevojshme për **v**izatimin e rrugës në **h**artën e Google-it, dhe që paraqet diferencën mes KPKP-së dhe KASP-së = KVH

Kohëzgjatje në sekonda e gjithë këtij **c**ikli = KC

Çdo kohë e raportuar në këtë raport të testimit real të sistemit është e formatit ora:minutat:sekondat:milisekondat

Vendet e përzgjedhura për vizitë

1. Prishtina; 42.6368896; 21.1148536
2. Skopje, Macedonia (FYROM); 41.9973462; 21.4279956
3. Tirana, Albania; 41.3275459; 19.8186982

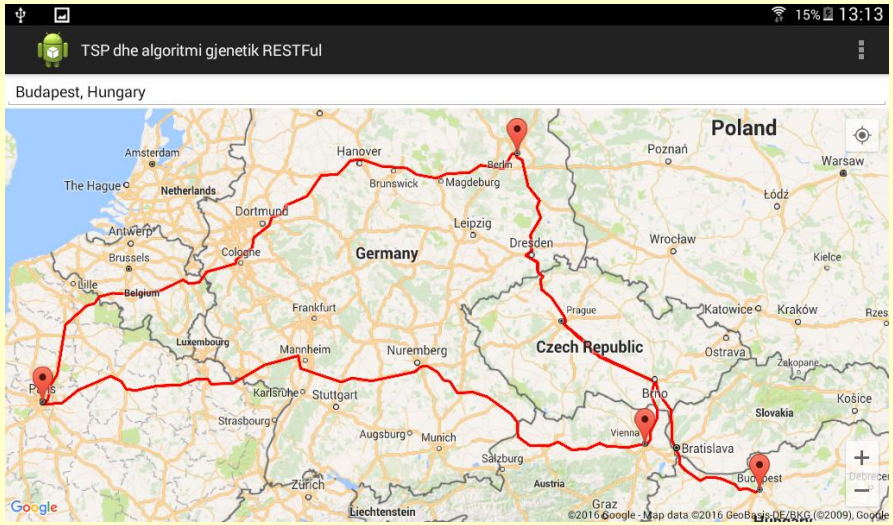
Rezultati nga serveri

1. Skopje, Macedonia (FYROM); 41.9973462; 21.4279956
2. Prishtina; 42.6368896; 21.1148536
3. Tirana, Albania; 41.3275459; 19.8186982
4. Prishtina; 42.6368896; 21.1148536

Kohët e shprehura në sekonda dhe milisekonda

KF: 12:25:28:556
KFA: 12:25:28:634
KPA: 12:25:28:704
KAGJ: 0.070
KASP: 12:25:29:216
D: 0.66
KPKP: 12:25:31:367
KVH: 2.151
KC: 2.811

Tabela 27: Testi I me 3 vende për t'u vizituar



Vendet e përzgjedhura për vizitë

1. Vienna, Austria; 48.2081743; 16.3738189
2. Berlin, Germany; 52.52000659999999; 13.404954
3. Paris, France; 48.85661400000001; 2.3522219
4. Budapest, Hungary; 47.497912; 19.040235

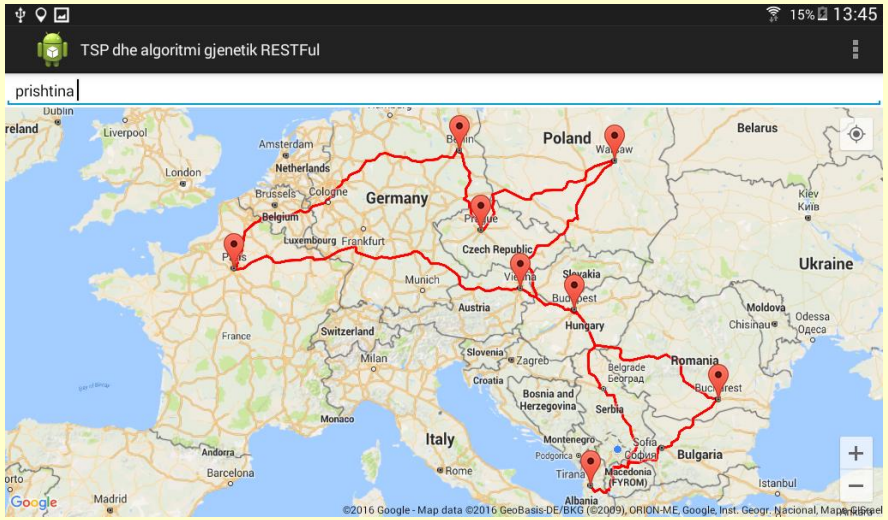
Rezultati nga serveri

1. Vienna, Austria; 48.2081743; 16.3738189
2. Budapest, Hungary; 47.497912; 19.040235
3. Berlin, Germany; 52.52000659999999; 13.404954
4. Paris, France; 48.85661400000001; 2.3522219
5. Vienna, Austria; 48.2081743; 16.3738189

Kohët e shprehura në sekonda dhe milisekonda

KF: 13:04:57:01
KFA: 13:04:57:561
KPA: 13:04:57:675
KAGJ: 1.298
KASP: 13:04:58:308
D: 1.208
KPKP: 13:05:01:775
KVH: 3.467
KC: 4.765

Tabela 28: Testi II me 4 vende për t'u vizituar



Vendet e përzgjedhura për t'u vizituar

1. Vienna, Austria; 48.2081743; 16.3738189
2. Berlin, Germany; 52.52000659999999; 13.404954
3. Paris, France; 48.85661400000001; 2.3522219
4. Prague, Czech Republic; 50.0755381; 14.4378005
5. Budapest, Hungary; 47.497912; 19.040235
6. Bucharest, Romania; 44.4267674; 26.1025384
7. Warsaw, Poland; 52.2296756; 21.0122287
8. Tirana, Albania; 41.3275459; 19.8186982

Rezultati nga serveri

1. Berlin, Germany; 52.52000659999999; 13.404954
2. Paris, France; 48.85661400000001; 2.3522219
3. Vienna, Austria; 48.2081743; 16.3738189
4; Budapest, Hungary; 47.497912; 19.040235
5. Tirana, Albania; 41.3275459; 19.8186982
6. Bucharest, Romania; 44.4267674; 26.1025384
7. Warsaw, Poland; 52.2296756; 21.0122287
8. Prague, Czech Republic; 50.0755381; 14.4378005
9. Berlin, Germany; 52.52000659999999; 13.404954

Kohët e shprehura në sekonda dhe milisekonda

KF: 13:40:55:359
KFA: 13:40:56:606
KPA: 13:40:56:669
KAGJ: 0.063
KASP: 13:04:57:445
D: 2.086
KPKP: 13:41:05:921
KVH: 8.477
KC: 10.563

Tabela 29: Testi III me 8 vende për t'u vizituar

Vendet e përzgjedhura për t'u vizituar

1. Madrid, Spain; 40.4167754; -3.7037902
2. Paris, France; 48.85661400000001; 2.3522219
3. Monaco; 43.73841760000001; 7.424615799999999
4. Zürich, Switzerland; 47.3768866; 8.541694
5. Vienna, Austria; 48.2081743; 16.3738189
6. Berlin, Germany; 52.52000659999999; 13.404954
7. Budapest, Hungary; 47.497912; 19.040235
8. Bucharest, Romania; 44.4267674; 26.1025384
9. Slovenia; 46.15124099999999; 14.995463
10. Montenegro; 42.70867799999999; 19.37439
11. Tirana, Albania; 41.3275459; 19.8186982
12. Prishtina; 42.6368896; 21.1148536

Rezultati nga serveri

1. Slovenia; 46.15124099999999; 14.995463
2. Monaco; 43.73841760000001; 7.424615799999999
3. Madrid, Spain; 40.4167754; -3.7037902
4. Paris, France; 48.85661400000001; 2.3522219
5. Zürich, Switzerland; 47.3768866; 8.541694
6. Berlin, Germany; 52.52000659999999; 13.404954
7. Vienna, Austria; 48.2081743; 16.3738189
8. Budapest, Hungary; 47.497912; 19.040235
9. Bucharest, Romania; 44.4267674; 26.1025384
10. Prishtina; 42.6368896; 21.1148536
11. Tirana, Albania; 41.3275459; 19.8186982
12. Montenegro; 42.70867799999999; 19.37439

Kohët e shprehura në sekonda dhe milisekonda

KF: 06:20:51:126
KFA: 06:20:53:798
KPA: 06:20:53:953
KAGJ: 0.155
KASP: 06:20:54:217
D: 3.091
KPKP: 06:21:08:83
KVH: 13.867
KC: 16.958

Tabela 30: Testi IV me 12 vende për t'u vizituar

Vendet e përzgjedhura për vizitë

1. Thessaloniki, Greece; 40.6400629; 22.9444191
2. Skopje, Macedonia (FYROM); 41.9973462; 21.4279956
3. Prishtina, District of Pristina; 42.6629138; 21.1655028
4. Tirana, Albania; 41.3275459; 19.8186982
5. Warsaw, Poland; 52.2296756; 21.0122287
6. Sofia, Bulgaria; 42.69770819999999; 23.3218675
7. Bratislava, Slovakia; 48.1485965; 17.1077477
8. Slovenia; 46.15124099999999; 14.995463
9. Monaco; 43.73841760000001; 7.424615799999999
10. Vienna, Austria; 48.2081743; 16.3738189
11. Berlin, Germany; 52.52000659999999; 13.404954
12. Zürich, Switzerland; 47.3768866; 8.541694
13. Budapest, Hungary; 47.497912; 19.040235
14. Montenegro; 42.70867799999999; 19.37439
15. Paris, France; 48.85661400000001; 2.3522219
16. Madrid, Spain; 40.4167754; -3.7037902

Rezultati nga serveri

1. Madrid, Spain; 40.4167754; -3.7037902
2. Paris, France; 48.85661400000001; 2.3522219
3. Berlin, Germany; 52.52000659999999; 13.404954
4. Warsaw, Poland; 52.2296756; 21.0122287
5. Vienna, Austria; 48.2081743; 16.3738189
6. Bratislava, Slovakia; 48.1485965; 17.1077477
7. Budapest, Hungary; 47.497912; 19.040235
8. Montenegro; 42.70867799999999; 19.37439
9. Tirana, Albania; 41.3275459; 19.8186982
10. Thessaloniki, Greece; 40.6400629; 22.9444191
11. Sofia, Bulgaria; 42.69770819999999; 23.3218675
12. Shkup, Skopje, Macedonia (FYROM); 41.9973462; 21.4279956
13. Prishtina, District of Pristina; 42.6629138; 21.1655028
14. Slovenia; 46.15124099999999; 14.995463
15. Zürich, Switzerland; 47.3768866; 8.541694
16. Monaco; 43.73841760000001; 7.424615799999999

Kohët e shprehura në sekonda dhe milisekonda

KF: 06:47:38:856
KFA: 06:47:40:389
KPA: 06:47:40:481
KAGJ: 0.092
KASP: 06:47:40:738
D: 1.882
KPKP: 06:47:54:912
KVH: 14.174
KC: 16.056

Tabela 31: Testi V me 16 vende për t'u vizituar

Interpretimi i rezultateve

Para se të filloj me interpretimin e rezultateve të lartëshënuara e shoh të udhës që rezultatet më të rëndësishme t'i prezantoj në mënyrë tabelore dhe përmes një grafiku, kështu që ato do të bëhen më të qarta për lexuesin.

Do të marr për shqyrtim vetëm 3 parametra kohorë, më të rëndësishmit, që janë:

kohëzgjatja në sekonda apo milisekonda e punës së **a**lgoritmit **gj**enetik = **KAGJ**,

koha e nevojshme në sekonda apo milisekonda për **v**izatimin e rrugës në **h**artën e Google-it = **KVH**, dhe

kohëzgjatja në sekonda e gjithë (komplet) **c**iklit = **KC**

	KAGJ	**KVH**	**KC**
3 vende për vizitë	0.07	2.151	2.811
4 vende për vizitë	1.298	3.467	4.765
8 vende për vizitë	0.063	8.477	10.563
12 vende për vizitë	0.155	13.867	16.958
16 vende për vizitë	0.092	14.174	16.056

Tabela 32: Parametrat kryesorë që ndikojnë në kohëzgjatjen e sistemit (kohëzgjatja në sekonda)

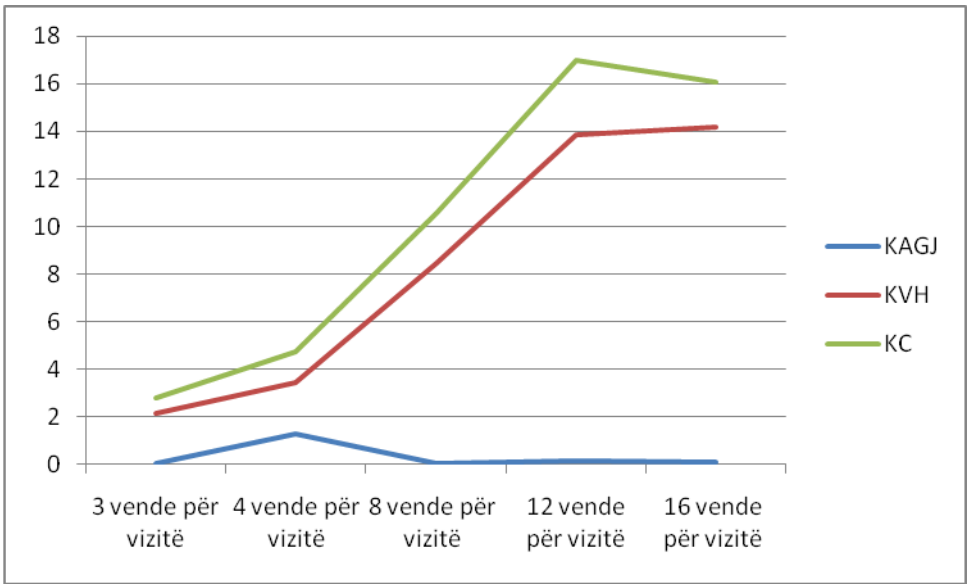


Figura 22: Parametrat kryesorë që ndikojnë në kohëzgjatjen e sistemit (grafikisht)

Siç mund të shihet nga tabela dhe nga grafiku i mësipërm, rritja e numrit të vendeve për t'u vizituar nuk ndikon aspak në kohëzgjatjen e ekzekutimit të algoritmit gjenetik. Me rritjen e numrit të vendeve për t'u vizituar, kjo kohë rritet në mënyrë të përshkallëzuar, mirëpo mund të themi se ky përshkallëzim është pothuajse i papërfillshëm meqë rritet me të qindtat e sekondës. Gjithashtu vlen të theksohet se kohëzgjatja e ekzekutimit të algoritmit është variabël në varësi të performancës së serverit ku algoritmi gjenetik ofrohet si ueb shërbim, si dhe nga ngarkesa e këtij serveri në kohën kur konsumohet ky ueb shërbim. Me ngarkesën e serverit ku ofrohet algoritmi si ueb shërbim nënkuptohet se sa klientë në të njëjtën kohë dërgojnë kërkesë për konsumimin e këtij ueb shërbimi.

Nga tabela dhe nga grafiku i mësipërm duket qartë se KVH është parametri që ndikon më së shumti në kohëzgjatjen e ekzekutimit të të gjithë sistemit. Pra, ky parametër përfaqëson kohën e nevojshme, që i nevojitet aplikacionit të Androidit për t'ia vizatuar

rrugën e saktë vizitorit në hartën e Google-it përmes polylines. Pra, është rruga që duhet ndjekur realisht nga vizitori.

Me rritjen e numrit të vendeve për t'u vizituar koha e ekzekutimit rritet në mënyrë drastike, meqë për çdo dy vende për t'u vizituar duhet kontaktuar ueb shërbimi i Google-it DIRECTIONS, në mënyrë që të marrim rrugën e saktë që lidh këto dy vende. Rezultati që kthehet nga serveri i Google-it duhet të përpunohet dhe mandej kjo rrugë e pjesshme duhet të vizatohet përmes plylines në hartë. Ky veprim kryhet në mënyrë të përsëritur deri në përmbylljen e turit. Ndonjëherë mund të ndodhë, që kur kërkojmë nga serveri i Google-it rrugën e saktë që lidh dy vendeve, ai propozon një rrugë që të çon përmes një vendi tjetër te vendi i dëshiruar (jo drejtpërdrejt). Kjo ndodhë nëse në realitet nuk ka rrugë që të çon drejtpërdrejt nga A në B, por vetëm përmes një vendi tjetër.

Rritja në mënyrë drastike varet, po ashtu, edhe nga gjatësia e turit përfundimtar. Kjo d. m. th.: se sa larg janë vendet që përbëjnë turin. Pra, pajisjes ku është i instaluar aplikimi i Androidit i duhet më shumë kohë për vizatimin përmes polylines të një rruge më të gjatë sesa të një rruge më të shkurtër. Po ashtu vlen të theksohet se kjo kohëzgjatje është e ndryshueshme varësisht nga ngarkesa e serverit të Google-it, nga shpejtësia e internetit, si dhe nga shpejtësia e pajisjes për përpunimin e të dhënave.

Nga tabela dhe grafiku i mësipërm na bëhet e qartë se nëse i mbledhim këta dy parametra (KGJA + KVH), shuma e tyre nuk jep KC (kohën e përgjithshme të ekzekutimit të komplet sistemit). Kjo lë të kuptohet që edhe parametra (faktorë) të tjerë kontribuojnë në kohëzgjatjen e ekzekutimit të komplet sistemit.

Këta faktorë janë:

a) Ngarkesa e wirelessroot-erit në të cilën është lidhur pajisja e vizitorit, pra numri i pajisjeve që janë lidhur në të. Këtu ndahet shpejtësia e internetit në përporcion me pajisjet që janë të lidhura në këtë root-er.
b) Shpejtësia e pajisjes për përpunimin e të dhënave të vendeve që ka përzgjedhur vizitori për t'i vizituar, nga harta e Google-i, paketimi i tyre në formatin JSON dhe dërgimi i tyre përmes internetit në server ku ofrohet ueb shërbimi. Pra, një nga faktorët është kohëzgjatja e dërgimit të shënimeve nga pajisja në server.
c) Koha që i duhet serverit për përshtatjen e shënimeve për përpunim të mëtejshëm në të. Po ashtu koha që i duhet serverit për paketimin e shënimeve në formatin JSON, pasi t'i ketë përpunuar ato.
d) Koha e dërgimit të shënimeve të përpunuara nga serveri te pajisja që dërgoj kërkesën për konsumimin e këtij ueb shërbimi. Pra, kohëzgjatja e kthimit të përgjigjes që përfshin kohën e përpunimit të kërkesës dhe kohën e kthimit të përgjigjes.

KAPITULLI VI. PËRFUNDIME/REKOMANDIME

Në këtë tezë doktorature janë programuar dy versione të algoritmit gjenetik si ueb shërbim, të cilët sa u përket funksioneve bazë, nuk dallojnë nga njëri -tjetri, mirëpo dallimi i vetëm është te programimi i funksionit fitness (funksioni për llogaritjen e gjatësisë së turit).

Te versioni i parë, funksioni fitness është programuar duke zbatuar formulën e Haversinës, për llogaritjen e gjatësisë së turit. Te versioni i dytë i algoritmit, funksioni për llogaritjen e gjatësisë së turit përdor ueb shërbimin e Google-it, të quajtur Google Distance Matrix.

Secili nga këto dy algoritme ka përparësitë dhe mangësitë e veta. Implementimi i formulës së Haversinës e bën algoritmin gjenetik që të jetë më i shpejtë në kryerjen e punës së tij, mirëpo bën që ai të marrë vendime jo të duhura, meqë në disa raste nuk vlerësohet korrekt gjatësia e turit. Pra, kjo formulë nuk merr parasysh situatën reale në terren, pra nëse ekziston rrugë direkte që lidh dy vende apo dy vendet lidhen përmes një rruge që kalon nëpër një vendi të tretë të ndërmjetëm (që në fakt është më e gjatë sesa gjatësia e llogaritur). Versioni i dytë i algoritmit, i cili shfrytëzon ueb shërbimin e Google-it për llogaritjen e gjatësisë së turit është më i ngadalshëm në kryerjen e punës së tij, meqë për çdo llogaritje duhet konsumuar ky shërbim në distancë. Pra, koha e kthimit të përgjigjes nga serveri i Google-it dhe koha e përpunimit të përgjigjes së kthyer janë dy shkaqet që e bëjnë algoritmin më të ngadaltë në kryerjen e punës së tij. Nga ana tjetër ky algoritëm merr vendimet e duhura, meqë gjatësia e tureve llogaritet saktë, duke u bazuar në distancat reale të vendeve në terren.

Me këto probleme ballafaqohet njeriu nëse dëshiron që algoritmin gjenetik ta përdorë për zgjidhjen e problemeve konkrete siç është rasti me biznesmenin udhëtar (TSP), ndërsa vetëm për kontribut akademik mjafton programimi i një versioni ku zbatohet formula e Teoremës së Pitagorës, që llogarit distancën e dy pikave në rrafsh (këtu punohet me koordinatat X,Y dhe jo me gjatësi dhe gjerësi gjeografike).

Algoritmi gjenetik fillon kërkimin e zgjidhjes optimale (në rastin TSP) prej një bashkësie të zgjidhjeve të mundshme, të quajtura popullatë, e jo prej një zgjidhjeje.

Algoritmi gjenetik përdor për kërkim vetëm një funksion (funksionin për vlerësim), për të dalluar zgjidhjen e mirë nga ajo e keqja (e papërdorshmja). Te ky funksion nuk vendosen kërkesa shtesë për të dalluar zgjidhjet.

Algoritmi gjenetik është i zbatueshëm për zgjidhjen e problemit TSP dhe ka ndikim të madh në optimizimin e rrugës së biznesmenit udhëtar deri në 746.499 %, bazuar në rezultatet e simulatorit për testim.

Algoritmi gjenetik është i zbatueshëm për zgjidhjen e të gjitha problemeve tek të cilat dështon gjetja e zgjidhjes ekzakte kombinatorike, pa pasur nevojë të ndryshohet shumë skeleti (bazamenti) i algoritmit.

Duhet pasur kujdes në aplikimin e algoritmit gjenetik, sepse ai është i aplikueshëm vetëm tek ato probleme për të cilat jemi në gjendje t'i gjenerojmë disa zgjidhje fillestare, meqë puna e tij mbështetet te këto zgjidhje dhe vazhdon me optimizim (duke i kombinuar ato dhe duke gjeneruar nga to zgjidhje të reja).

Madhësia e popullatës fillestare duhet të jetë sa më e madhe në mënyrë që të përftojmë rezultat më optimal, mirëpo duhet pasur parasysh që kohëzgjatja e ekzekutimit të algoritmit në këtë rast rritet, bazuar në rastet për testim dhe te rezultatet e përftuara përmes simulatorit të programuar në JAVA.

Një nga objektivat e kësaj teze doktorale ishte edhe përzgjedhja e një algoritmi tjetër joekzakt për krahasim. Këtu është përzgjedhur algoritmi: Simulated Annealing, principi i punës të të cilit është po ashtu simulimi i një procesi natyror në kompjuter. Përmes një simulatori të programuar në JAVA është testuar, po ashtu, puna e tij gjatë zgjidhjes së problemit TSP. Përmes këtij simulatori është simuluar puna e tij grafikisht dhe po ashtu është testuar performanca e tij, përfshi këtu kohën e ekzekutimit si dhe rrugën e optimizuar. Bazuar në rezultatet e përftuara, algoritmi Simulated Annealing është më efikas në optimizimin e rrugës, ndërsa koha e ekzekutimit të tij është shumë më e shkurtër sesa ajo e algoritmit gjenetik për numër të njëjtë vendesh për t'u vizituar.

Të dyja algoritmet kanë ngjashmëri dhe dallime.

Ngjashmëritë janë:

1) Principi i punës së tyre bazohet në simulimin e proceseve natyrore në kompjuter: Algoritmi gjenetik simulon përzgjedhjen natyrore ku më i forti mbijeton, ndërsa simulated annealing simulon përzgjedhjen e procesit të shkrirjes së metaleve në metalurgji.
2) Që të dyja algoritmet japin zgjidhje të përafërt për TSP

3) Që të dyja algoritmet janë të zbatueshme për problemet për të cilat jemi në gjendje të gjenerojmë zgjidhje fillestare

Dallimi mes këtyre algoritmeve është se algoritmi Simulated Annealing optimizon më tepër rrugën e biznesmenit udhëtar dhe si rrjedhojë rezultati është më optimal, më i kënaqshëm dhe i afrohet më tepër rezultatit ekzakt.

Vetëm falë programimit të simulatorëve në JAVA për testim dhe demonstrim grafik të punës së këtyre algoritmeve, është bërë e mundur testimi i përqindjes së optimizimit të rrugës së biznesmenit udhëtar si dhe krahasimi i këtyre dy algoritmeve për kah performanca.

Përmes programimit të një aplikacioni të Androidit është bërë integrimi i ueb shërbimeve të Google-it, respektivisht hartave të Google-it, zgjidhje këto që iu shoqëruan zgjidhjes së problemit biznesmeni udhëtar. Në këtë aplikacion Androidi janë integruar ueb shërbimet e Google-it në mënyrë që përdoruesit t'i ofrohet një ndërfaqe sa më interaktive me aplikacionin, kështu që ai ta ketë të lehtë dhe në mënyrë intuitive përdorimin e tij. Po ashtu përmes konsumimit të algoritmit gjenetik si ueb shërbim nga ana e këtij aplikacioni të Androidit, është bërë i mundur optimizimi i rrugës së biznesmenit udhëtar. Pra, ky aplikacion është klient i dyfishtë meqë shfrytëzon ueb shërbimet e Google-it në njërën anë dhe algoritmin si ueb shërbim në anën tjetër. Përdoruesit i vizatohet rruga që duhet ndjekur në hartën e Google-it, si dhe përmes një dritareje për informim, ai informohet se sa kilometra është gjatësia e turit që lidh vendet e përzgjedhura për t'u vizituar.

6.1. Puna në të ardhmen

Puna në të ardhmen, e cila nuk ka qenë objektiv i kësaj teze doktorate, është që algoritmi gjenetik të mos ofrohet vetëm si ueb shërbim RESTful, por edhe si SOAP, si dhe parametrat që ndikojnë punën e algoritmit gjenetik të jenë të konfigurueshëm nga ana e përdoruesit. Nëse këta parametra nuk jepen nga ana e shfrytëzuesit të këtij ueb shërbimi, t'i ketë ueb shërbimi default, ndërsa nëse jepen të merren parasysh. Parametrat opcionalë për të cilët bëhet fjalë janë: madhësia e popullatës fillestare, numri i gjeneratave që riprovohen për të optimizuar rezultatin si dhe kohëzgjatja e ekzekutimit të algoritmit. Një gjë tjetër me rëndësi do të ishte që përdoruesit t'i lihej mundësia që të zgjedhë edhe llojin e formatit të përgjigjes. Pra, së bashku me parametrat e tjerë të programohet që përdoruesi, përmes një parametri tjetër, të zgjedhë formatin e përgjigjes nga serveri: XML apo JSON, meqë për momentin kthehet vetëm përgjigjja e formatit JSON.

Unë po planifikoj, po ashtu, që në të ardhmen të programoj një algoritëm hibrid që do të jetë kombinim i algoritmeve ekzakte (brute force dhe greedy), algoritmit gjenetik dhe simulated annealing, në mënyrë që t'i japë inteligjencë më të madhe artificiale algoritmit të modifikuar. Ky algoritëm le të jetë më inteligjent në kuptimin që, nëse numri i vendeve për t'u vizitë është më i vogël se 10, të vendosë vetvetiu të zbatojë algoritmin brute force, nëse është 10 – 30 të zbatojë algoritmin greedy për zgjidhjen e problemit TSP, 31 deri në 100 algoritmin gjenetik dhe mbi 100 vende për t'u vizituar të hyjë në punë algoritmi simulated annealing. Këtë do ta bëj bazuar në rezultatet e përfituara nga testimi i algoritmeve në punimet e ngjashme. Nëse përdoruesi përzgjedh deri në 10 vende për vizitë ai do të marrë si rezultat turin më optimal, meqë puna e algoritmit brute force nuk ndikohet shumë nga ky

numër i vendeve për t'u vizituar. Nga 2 deri në 30 vende për t'u vizituar koha e pritur e ekzekutimit të algoritmeve ekzakte është e perceptueshme, ndërsa mbi 30 vende për t'u vizituar zbatohen algoritmet apo metodat e përafërta. Arsyeja pse bëj dallim në mes të metodave të përafërta është se mbi 100 vende për t'u vizituar algoritmi simulated annealing e merr primatin ndaj algoritmit gjenetik. Pra, bazuar në rezultatet e simulaturëve për testimin e të dyja këtyre algoritmeve, algoritmi gjenetik optimizon më tepër turin dhe kohëzgjatja e ekzekutimit të tij është më e shkurtër. Si rezultat i kësaj përdoruesi në pajisjen mobile merr turin më të shkurtër dhe për një kohë më të shkurtër sesa koha e ekzekutimit të sistemit të tanishëm.

SHTOJCAT

Shtojca A: Klasa JsonReader

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.HttpEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import android.app.Activity;

import java.io.BufferedReader;
import java.io.IOException;
```

```
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.UnsupportedEncodingException;
import java.net.URL;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.util.Map;
import java.util.Map.Entry;
import java.util.List;

import org.json.JSONArray;
import org.json.JSONObject;
import org.json.JSONException;


public class JsonReader {

    private static String readAll(final Reader rd) throws IOException {
        final StringBuilder sb = new StringBuilder();
        int cp;
        while ((cp = rd.read()) != -1) {
            sb.append((char) cp);
        }
        return sb.toString();
    }
```

```
    public static JSONObject read(final String url) throws IOException, JSONException
{
        final InputStream is = new URL(url).openStream();
        try {
            final BufferedReader rd = new BufferedReader(new InputStreamReader(is,
Charset.forName("UTF-8")));
            final String jsonText = readAll(rd);
            final JSONObject json = new JSONObject(jsonText);
            return json;
        } finally {
            is.close();
        }
    }

    public static JSONObject getJSONfromURL(String url){
        InputStream is = null;
        String result = "";
        JSONObject jArray = null;
        //http post
        try{
            HttpClient httpclient = new DefaultHttpClient();
            HttpGet httppost = new HttpGet(url);
            HttpResponse response = httpclient.execute(httppost);
            HttpEntity entity = response.getEntity();
            is = entity.getContent();
        }catch(Exception e){
            Log.e("log_tag", "Error in http connection "+e.toString());
        }
```

```
        //convert response to string
        try{
            BufferedReader reader = new BufferedReader(new InputStreamReader(is,"iso-
8859-1"),8);
            StringBuilder sb = new StringBuilder();
            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
            is.close();
            result=sb.toString();
            throw new Exception(result);
        }catch(Exception e){
            Log.e("log_tag", "Error converting result "+e.toString());
        }
       //try parse the string to a JSON object
       try{
            jArray = new JSONObject(result);
       }catch(JSONException e){
            Log.e("log_tag", "Error parsing data "+e.toString());
       }
       return jArray;
    }


    public static String join(List<?> list, String delim) {
        int len = list.size();
        if (len == 0)
```

```
            return "";
        StringBuilder sb = new StringBuilder(list.get(0).toString());
        for (int i = 1; i < len; i++) {
            sb.append(delim);
            sb.append(list.get(i).toString());
        }
        return sb.toString();
    }

}
```

Shtojca B: Klasa Vendi

```
import java.nio.charset.Charset;
import org.json.JSONArray;
import org.json.JSONObject;
import org.json.JSONException;
import java.lang.Math.*;

public class Vendi {
    private String id;
    private String emri;
    private String adresa;
    private double gjeresia_gjeografike;
    private double gjatesia_gjeografike;
    private static Double RADIUSI_TOKES = 6371.00;
```

```
    public Vendi() {
    }
    public Vendi(String id, String emri, String adresa, double gjeresia_gjeografike,
double gjatesia_gjeografike) {
        this.id = id;
        this.emri = emri;
        this.adresa = adresa;
        this.gjeresia_gjeografike = gjeresia_gjeografike;
        this.gjatesia_gjeografike = gjatesia_gjeografike;
    }

    public void vendosID(String id) {
        this.id = id;
    }

    public String thirrID() {
        return id;
    }

    public void vendosEmrin(String emri) {
        this.emri = emri;
    }

    public String thirrEmrin() {
        return emri;
    }

    public void vendosAdresen(String adresa) {
```

```
    this.adresa = adresa;
}

public String thirrAdresen() {
    return adresa;
}

public void vendosGjeresine(double gjeresia_gjeografike) {
    this.gjeresia_gjeografike = gjeresia_gjeografike;
}

public double thirrGjeresine() {
    return gjeresia_gjeografike;
}

public void vendosGjatesine(Double gjatesia_gjeografike) {
    this.gjatesia_gjeografike = gjatesia_gjeografike;
}

public double thirrGjatesine() {
    return gjatesia_gjeografike;
}

@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append(id);
    sb.append(',');
```

```
        sb.append(emri);
        sb.append(',');
        sb.append(adresa);
        sb.append(',');
        sb.append(gjeresia_gjeografike);
        sb.append(',');
        sb.append(gjatesia_gjeografike);
        return sb.toString();
    }


    public Vendi ktheObjektin() {
        return this;
    }

    public final Vendi jsonNeObjekt(JSONObject objekti) throws JSONException {
        String id = objekti.getString("id");
        String emri = objekti.getString("emri");
        String adresa = objekti.getString("adresa");
        double gjeresia = objekti.getDouble("gjeresia_gjeografike");
        double gjatesia = objekti.getDouble("gjeresia_gjeografike");
        return new Vendi(id, emri, adresa, gjeresia, gjatesia);
    }

    public JSONObject konvertoNeJSON(Vendi vendi) {
        try {
            JSONObject objekti = new JSONObject();
            objekti.put("id", vendi.thirrID());
```

```
            objekti.put("emri", vendi.thirrEmrin());
            objekti.put("adresa", vendi.thirrAdresen());
            objekti.put("gjeresia_gjeografike", vendi.thirrGjeresine());
            objekti.put("gjatesia_gjeografike", vendi.thirrGjatesine());
            return objekti;
        } catch(JSONException ex) {
            ex.getMessage();
        }
        return null;
    }

    public static Double NeRadian(Double shkalla) {
        return shkalla * Math.PI / 180;
    }

    public double largesiaDeri(Vendi vendi) {
        double dGjeresi = NeRadian(vendi.gjeresia_gjeografike - gjeresia_gjeografike);
        double dGjatesi = NeRadian(vendi.gjatesia_gjeografike - gjatesia_gjeografike);
        double gjeresia1 = NeRadian(gjeresia_gjeografike);
        double gjeresia2 = NeRadian(vendi.gjeresia_gjeografike);
        double a = Math.pow(Math.sin(dGjeresi/2), 2) + Math.pow(Math.sin(dGjatesi/2),
2) *
                Math.cos(gjeresia1) * Math.cos(gjeresia2);
        double c = 2.0*Math.atan2(Math.sqrt(a), Math.sqrt(1.0 - a));
        return RADIUSI_TOKES * c;
    }
}
```

Shtojca C: GooglePlacesAutocompleteAdapter

```
public class GooglePlacesAutocompleteAdapter extends ArrayAdapter {

     private ArrayList resultList;
     private static final String LOG_TAG = "Google Places Autocomplete";
     private static final String PLACES_API_BASE =
"https://maps.googleapis.com/maps/api/place";
     private static final String TYPE_AUTOCOMPLETE = "/autocomplete";
     private static final String OUT_JSON = "/json";
     private static String key = "MY_KEY";


     public GooglePlacesAutocompleteAdapter(Context context, int textViewResourceId) {
          super(context, textViewResourceId);
     }

     public void onItemClick(AdapterView adapterView, View view, int position, long
id) {
          String str = (String) adapterView.getItemAtPosition(position);
          Toast.makeText(this.getContext(), str, Toast.LENGTH_SHORT).show();
     }

     @Override
     public int getCount() {
```

```
     return resultList.size();
}

@Override
public String getItem(int index) {
     return (String) resultList.get(index);
}

@Override
public Filter getFilter() {

     Filter filter = new Filter() {

     @Override
     protected FilterResults performFiltering(CharSequence constraint) {
               FilterResults filterResults = new FilterResults();
            if (constraint != null) {
               // Retrieve the autocomplete results.
               resultList = MainActivity.autocomplete(constraint.toString());
               // Assign the data to the FilterResults
               filterResults.values = resultList;
                filterResults.count = resultList.size();
             }
               return filterResults;
        }


     @Override
```

```
          protected void publishResults(CharSequence constraint, FilterResults
results) {
               if (results != null && results.count > 0) {
                    notifyDataSetChanged();
               } else {
                    notifyDataSetInvalidated();
               }
          }

          };

          return filter;
     }


     public static ArrayList autocomplete(String input) {
          ArrayList resultList = null;

          HttpURLConnection conn = null;
          StringBuilder jsonResults = new StringBuilder();
          try {
               StringBuilder sb = new StringBuilder(PLACES_API_BASE +
TYPE_AUTOCOMPLETE + OUT_JSON);
               sb.append("?key=" + key);
               //sb.append("&components=country:gr");
               sb.append("&input=" + URLEncoder.encode(input, "utf8"));

               URL url = new URL(sb.toString());
```

```
            conn = (HttpURLConnection) url.openConnection();
            InputStreamReader in = new InputStreamReader(conn.getInputStream());

            // Load the results into a StringBuilder
            int read;
            char[] buff = new char[1024];
            while ((read = in.read(buff)) != -1) {
                jsonResults.append(buff, 0, read);
            }
        } catch (MalformedURLException e) {
        Log.e(LOG_TAG, "Error processing Places API URL", e);
        } catch (IOException e) {
        Log.e(LOG_TAG, "Error connecting to Places API", e);
        } finally {
            if (conn != null) {
                conn.disconnect();
            }
            return resultList;
        }
    }
}
```

Shtojca D: PlaceAdapter

```
import java.io.IOException;
import java.io.InputStreamReader;
```

```
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;
import java.util.ArrayList;

import android.app.Activity;
import android.content.Context;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Filter;
import android.widget.TextView;
import android.widget.Toast;

public class PlaceAdapter extends ArrayAdapter<GooglePlace> {

     private static class ViewHolder {
           TextView placeid;
           TextView name;
     }

     public PlaceAdapter(Context context, ArrayList<GooglePlace> places) {
           super(context, R.layout.item_place, places);
           // TODO Auto-generated constructor stub
```

```
     }

      @Override
      public View getView(int position, View convertView, ViewGroup parent) {
           GooglePlace place = getItem(position);
           if (convertView == null) {
                convertView =
LayoutInflater.from(getContext()).inflate(R.layout.item_place, parent, false);
           }
           TextView id = (TextView) convertView.findViewById(R.id.placeid);
           TextView name = (TextView) convertView.findViewById(R.id.placename);
           id.setText(place.thirrID());
           name.setText(place.thirrEmrin());
           return convertView;
       }
}
```

Shtojca E: Directions dhe polylines

```
     long distanceForSegment = 0;
     try {
          JsonReader jsonClass = new JsonReader();
             StringBuilder builder = new StringBuilder();
             builder.append(url_directions).append("origin=").append(prej);
             builder.append("&destination=").append(deri);
             builder.append("&waypoints=optimize:true|");
             builder.append("&sensor=true");
```

```
            final JSONObject pergjigja = jsonClass.read(builder.toString());
            JSONArray rezultati = pergjigja.getJSONArray("routes");
            distanceForSegment =
rezultati.getJSONObject(0).getJSONArray("legs").getJSONObject(0).getJSONObject("distan
ce").getInt("value");
            JSONArray steps =
rezultati.getJSONObject(0).getJSONArray("legs").getJSONObject(0).getJSONArray("steps")
;
            List<LatLng> lines = new ArrayList<LatLng>();
           for(int i=0; i < steps.length(); i++) {
               String polyline =
steps.getJSONObject(i).getJSONObject("polyline").getString("points");
               for(LatLng p : decodePolyline(polyline)) {
                   lines.add(p);
               }
           }
           myMap.addPolyline(new
PolylineOptions().addAll(lines).width(3).color(Color.RED));

     } catch(Exception e) {
          Log.e("GABIMI --> ", "Response string buffer error: " + e.getMessage());
        }
     return distanceForSegment;

    }



private List<LatLng> decodePolyline(String encoded) {
        List<LatLng> poly = new ArrayList<LatLng>();
        int index = 0, len = encoded.length();
        int lat = 0, lng = 0;
```

```
    while (index < len) {
        int b, shift = 0, result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lat += dlat;
        shift = 0;
        result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lng += dlng;
        LatLng p = new LatLng((double) lat / 1E5, (double) lng / 1E5);
        poly.add(p);
    }
    return poly;
}


public void addPolylineToMap(List<LatLng> latLngs) {
 PolylineOptions options = new PolylineOptions();
    for (LatLng latLng : latLngs) {
      options.addAll(latLngs).width(3).color(Color.RED);
    }
    myMap.addPolyline(options);
}
```

REFERENCAT

(2006). Retrieved January 26, 2017, nga http://www.cs.yorku.ca/~aaw/Zambito/TSP_Survey.pdf

(2015, August). Retrieved January 30, 2017, nga Javarevisited: http://javarevisited.blogspot.com/2015/08/difference-between-soap-and-restfull-webservice-java.html

(2017, January 24). (APACHE) Retrieved March 1, 2017, nga Apache Tomcat: http://tomcat.apache.org/

(2017, January 12). (ORACLE) Retrieved March 01, 2017, nga https://netbeans.org/

Adewole, A., Otubamowo, K., & Egunjobi, T. (a.d.). A Comparative Study of Simulated Annealing and Genetic Algorithm for Solving the Salesman Problem. *International Journal of Applied Information Systems (IJAIS)* .

Alphred, H. (a.d.). Retrieved April 12, 2014, nga http://learntech.rwth-aachen.de/lehre/ss05/fdi/folien/VS11HTML/TSP.pdf

Android. (a.d.). (Google) Retrieved November 6, 2016, nga https://developers.google.com/maps/android/

Angel, R., Caudle, W., & Noonan, R. &. (1972). Computer assisted school bus scheduling. *Management Science* , 279–288.

Anupriya, & Saxena, M. (2013). An Android Application for Google Map Navigation System Implementing Travelling Salesman Problem. *International Journal of Computer & Organization Trends , III* (4).

Arora, K., & Mamta, A. (2016). Better Result for Solving TSP: GA versus ACO. *International Journal of Advanced Research in Computer Science and Software Engineering , VI* (3).

Babar, V., Jadhav, S., Jagtap, P., & Joshi, A. (2014). Implementation of an Application on Android Devices for Optimal Tour. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST) , II* (1).

Baun, C., Kunze, M., Nimis, J., & Tai, S. (2011). *Cloud Computing (Web-basierte dynamische IT-Services)* (bot. i 2nd). Berlin: Springer.

Benusi, A. (2015). Retrieved November 15, 2016, nga http://www.doktoratura.unitir.edu.al/wp-content/uploads/2015/09/Doktoratura-Ardi-Benusi-Fakulteti-i-Shkencave-i-Natyrore-Departamenti-i-Informatikes.pdf

Biggs, N., Lloyd, K., & Wilson, R. (1986). Graph Theory. (fv. 1736-1936). Clarendon Press.

Böhm, M., Leimeister, S., Riedl, C., & Krcmar, K. (2014, December 9). *Technische Universität München.* Retrieved March 12, 2016, nga https://www.researchgate.net/profile/Markus_Boehm/publication/268011245_Cloud_Computing_and_Computing_Evolution/links/548726750cf2ef34478ec2de.pdf

Bradley, S., Hax, A., & Magnanti, T. (1977). *Applied Mathematical Programming.* Addison-Wesley.

Brownlee, J. (2012). *Clever Algorithms Nature-Inspired Programming Recipes* (bot. i 2nd).

Ch. Chauhan, R. G., & Pathak, K. (2012). Survey of Methods of Solving TSP along with its Implementation using Dynamic Programming Approach. *International Journal of Computer Applications , LII* (4).

Cipeluch, B., Jacob, R., Winstanley, A., & Mooney, P. (2010). Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps. *Accuracy 2010 Symposium.* Leicester, UK.

Darshni, E. P., & Kaur, G. (2010). IMPLEMENTATION OF ACO ALGORITHM FOR EDGE DETECTION AND SORTING SALESMAN PROBLEM. *International Journal of Engineering Science and Technology , 6* (06), 2304-2315.

Dharmale, P. N., & Ramteke, P. L. (2015). Mobile Cloud Computing. *International Journal of Science and Research (IJSR) , IV* (1).

Die Betriebssysteme im Überblick. (2016, November 23). Retrieved January 1, 2017, nga Betriebssystem Android: https://www.test.de/Handys-und-Smartphones-im-Test-4222793-4222876/

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the traveling salesman problem. *BioSystems.* Belgium: Press.

EDUCUASE|Library. (a.d.). Retrieved November 26, 2016, nga Mobile Learning: https://library.educause.edu/topics/teaching-and-learning/mobile-learning

Fejzaj, J. (2014). Retrieved November 5, 2016, nga https://docs.google.com/a/fshn.edu.al/viewer?a=v&pid=sites&srcid=ZnNobi5lZHUuYWx8ZnNobnxneDoxYzA0NWJiYmQyZjlhOWZh

Forgaty International Center. (a.d.). Retrieved në https://www.fic.nih.gov/RESEARCHTOPICS/Pages/MobileHealth.aspx

Google Maps APIs. (a.d.). Retrieved April 30, 2016, nga https://developers.google.com/maps/

Grötschel, M., Jünger, M., & Reinelt, G. (1991). Optimal Control of Plotting and Drilling Machines: A Case Study. *Mathematical Methods of Operations Research , III* (1), 61-84.

Grubel, G. (2003, April). Retrieved May 03, 2016, nga https://www.ads.tuwien.ac.at/publications/bib/pdf/feltl_03.pdf

Häckel, S., & Lemke, S. (2012, December 18). *TU Chemnitz*. Retrieved April 12, 2014, nga https://www.tu-chemnitz.de/wirtschaft/wi2/lehre/2012WS/EUS/eus_teil2_anwendung.pdf

Halili, F. (2014). Retrieved November 6, 2016, nga https://drive.google.com/viewerng/a/fshn.edu.al/viewer?a=v&pid=sites&srcid=ZnNobi5lZHUuYWx8ZnNobnxneDo0YjRiMmViZjI1MDI5NTRi

Harbich, S. (2007, 12 26). *Einführung genetischer Algorithmen mit Anwendungsbeispiel*. Retrieved 11 05, 2016, nga http://www-e.uni-magdeburg.de/harbich/genetische_algorithmen/genetische_algorithmen.pdf

Hurwitz, J., Bloor, R., Kaufman, M., & Halper, F. (2009). *Cloud Computing For Dummies*. Paperback.

Ilhan, I. (2016). Solution for the Travelling Salesman Problem with a Microcontroller-based Instantaneous System. *Intelligent Systems and Applications in Engineering , IV* (4), 122-127.

Imbach, R. (a.d.). Retrieved November 11, 2016, nga https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwjet7aX_prQAhWIcRQKHSaOAgwQFggeMAA&url=http%3A%2F%2Fweb.fhnw.ch%2Fpersonenseiten%2Ftaoufik.nouri%2Fkbs%2FGenetischer%2520Algorithmus-imbach.pdf&usg=AFQjCNH2TfpvFAmgv2QbfYJ1K6qU0SBbP

J. K. Lenstra, A. H. (1975). Some Simple Applications of the Travelling Salesman Problem. *Opl. Res. Q , XXVI* (4), 717-733.

Jakob I. Bernoulli, B.-a.-B. (a.d.). Retrieved May 02, 2016, nga http://www.mathepedia.de/Branch-and-Bound.aspx

Kalavace, K. (2013). Retrieved November 28, 2016, nga https://www.academia.edu/6258439/Disponueshmeri_permes_virtualizimit

Kämpf, M. (2006, May). Retrieved 05 11, 2016, nga TU Chemnitz: http://www.qucosa.de/fileadmin/data/qucosa/documents/5310/data/CSR-06-04.pdf

Katayama, K., Sakamoto, S., & Narihisa, H. (2000). The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem. *International Journal of Mathematical and Computer Modelling , XXXI* (10-12), 197-203.

Lenstra, J. &. (1974). *Some Simple Applications of the Travelling Salesman Problem.* Amsterdam.

Marinescu, D. C. (2013). *Cloud Computing. Theory and Practice.* USA: Elsevier & Book Aid International.

Marshall, A. (2010). *Earthware*. Retrieved February 22, 2017, nga Mapping APIs - Google Maps vs Bing Maps: Part 2 Licensing: http://www.earthware.co.uk/blog/mapping-apis-google-maps-vs-bing-maps-part-2-licensing/#

Martín, S., Jessica, C., Ignacio, P., & Nélida, B. (2004). On Stopping Criteria for Genetic Algorithms. Në S. L. Ana L. C. Bazzan (Re.), *17th Brazilian Symposium on Artificial Intelligence. 3171 2004.* Sao Luis, Maranhao (Brazil): Springer.

Matai, R., Singh, S. P., & Mittal, M. L. (2010, November). *Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches.* Retrieved 11 05, 2016, nga ResearchGate: https://www.researchgate.net/publication/221909777

Mataija, M., Šegić, M. R., & Jozić, F. (2016). Solving the travelling salesman problem using the Branch and Bound method. *Zbornik Veleučilišta u Rijeci , IV* (1), 259-270.

Mell, P., & Grance, T. (2011, September). Retrieved November 8, 2016, nga http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Microsoft Virtual Server. (a.d.). Retrieved March 20, 2013, nga http://www.microsoft.com/windowsserversystem/virtualserver

Mobile Cloud Computing. (2015, March). Retrieved November 10, 2016, nga http://geo.edu.al/cloud/wp-content/uploads/2015/03/Mobile-Cloud-Computing.pdf

Musafi, S. (a.d.). *Kompjuterizimi në Re*. Retrieved November 10, 2016, nga http://fjalaime.ch/karakteristikat-e-kompjuterizimit-ne-re-cloud-computing/

Nodehi, N., Fadaei, M., & Ebrahimi, P. (2016). Solving the traveling salesman probleSolving the traveling salesman problem using randomized gravitational emulation searchm using randomized gravitational emulation search. *JOURNAL OF CURRENT RESEARCH IN SCIENCE , S* (2), 818-821.

Noraini, M. R., & Geraghty, J. (2011). Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. *The 2011 International Conference of Computational Intelligence and Intelligent Systems.* London.

Osaba, E., Yang, X.-S., Diaz, F., Lopez-Garcia, P., & Carballed, R. (2016). An Improved Discrete Bat Algorithm for Symmetric and Asymmetric Traveling Salesman Problems. *Engineering Applications of Artificial Intelligence , 48* (1), 59-71.

Paul Theodor Pyl, K. L., & Rudolph. (a.d.). Retrieved May 05, 2016, nga https://www.activevb.de/tutorials/tut_antalgo/downloads/Ameisenalgorithmus%20und%20TSP.pdf

Pëllumbi, I. (2014, Jul 11). *SlideShare*. Retrieved November 24, 2016, nga http://www.slideshare.net/pellumbimatilda/virtualizimi

Petrakis, E. (a.d.). Retrieved May 03, 2016, nga http://www.intelligence.tuc.gr/~petrakis/courses/datastructures/algorithms.pdf

Places API for Android . (a.d.). (Google) Retrieved May 15, 2016, nga https://developers.google.com/places/android-api/signup#release-cert

Places API Web Service . (a.d.). (Google) Retrieved May 15, 2016, nga https://developers.google.com/places/web-service/

Pothineni, C. (2013, Decembar 13). Retrieved January 25, 2017, nga http://cs.indstate.edu/cpothineni/alg.pdf

Reiter, S. (2016, November 28). *ASCENS Project Blo*. Retrieved në Dreaming of fluffy clouds: http://blog.ascens-ist.eu/2011/03/dreaming-of-fluffy-clouds/

Rubin, A. (2013, November 21). *Percona*. (MySQL AB) Retrieved November 11, 2016, nga https://www.percona.com/blog/2013/10/21/using-the-new-mysql-spatial-functions-5-6-for-geo-enabled-applications/

Sadeh, N. M. (2010). *Introduction to Mobile Commerce*. Retrieved November 29, 2016, nga http://mcom.cs.cmu.edu/

Saiyed, A. R. (2012, Aprill 11). Retrieved January 25, 2017, nga http://cs.indstate.edu/~zeeshan/aman.pdf

Saleh, H., & Chelouah, R. (2004). The design of the global navigation satellite system surveying networks using genetic algorithms. *Engineering Applications of Artificial Intelligence , XVII*, 112-122.

Stanec, R. (2011, May 01). Solving of Travelling Salesman Problem for large number of cities in environment with constraints.

Stefan, B. (2002, November 29). Retrieved April 14, 2014, nga https://www2.informatik.uni-erlangen.de/teaching/thesis/download/i2S00379.pdf

Tafa, I. (2013). Retrieved November 24, 2016, nga http://www.fti.edu.al/data/pages/47/attach/prezantimi_phd_igli_plote.pdf

Tianfield, H. (2012). Security Issues in Cloud Computing. *IEEE International Conference on Systems, Man, and Cybernetics.*

Tidwell, D., Snell, J., & Kulchenko, P. (2001). *Programming Web Services with SOAP* (bot. i 1st). O'Reilly.

T-Online. (2016). Retrieved January 1, 2017, nga Smartphones und Tablets: http://www.t-online.de/ratgeber/technik/handy/id_46530318/android-betriebssystem-vor-und-nachteile.html

Trautmann, R. (2016, August). *teltarif.de*. Retrieved January 1, 2017, nga Android: Googles Smartphone-System hat den Markt erobert: https://www.teltarif.de/handy/betriebssysteme/android.html

TWT. (2015, November 15). Retrieved February 22, 2017, nga https://www.twt.de/news/detail/google-maps-vs-openstreetmap-welcher-kartendienst-ist-der-beste.html

Vishnupriyan, S., Govindarajan, L., Prabhakaran, G., & Ramachandran, K. Quality Improvement in Higher Education through Normalization of Student Feedback data using Evolutionary Algorithm. *International Journal of Applied Management and Technology , VI* (3).

w3schools, XML DTD. (a.d.). Retrieved April 3, 2016, nga http://www.w3schools.com/xml/xml_dtd.asp

w3schools, XML Namespaces. (a.d.). Retrieved April 3, 2016, nga http://www.w3schools.com/xml/xml_namespaces.asp

w3schools, XML Schema. (a.d.). Retrieved April 3, 2016, nga http://www.w3schools.com/xml/xml_schema.asp

w3schools, XML Soap. (a.d.). Retrieved May 2, 2016, nga http://www.w3schools.com/xml/xml_soap.asp

w3schools, XML WSDL. (a.d.). Retrieved April 5, 2016, nga http://www.w3schools.com/xml/xml_wsdl.asp

Web Services. (a.d.). (Google) Retrieved April 25, 2016, nga https://developers.google.com/maps/web-services/overview#

Web Services, Directions API. (a.d.). (Google) Retrieved April 26, 2016, nga https://developers.google.com/maps/documentation/directions/intro#Introduction

Web Services, Distance Matrix API. (a.d.). (Google) Retrieved April 25, 2016, nga https://developers.google.com/maps/documentation/distance-matrix/intro

Web Services, Geocoding API. (a.d.). (Google) Retrieved May 04, 2016, nga https://developers.google.com/maps/documentation/geocoding/intro

Yang, F. (2010, May 18). Retrieved November 05, 2016, nga http://courses.csail.mit.edu/18.337/2010/projects/reports/Fan_Yang_Final_Project_Report_Fixed_Figure_References.pdf

Yang, X.-S. (2010). *A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization.* Springer.

ZENUNI, V. (2007). Retrieved November 6, 2016, nga http://fjalaime.ch/wp-content/uploads/Puinim-Diplome-Ueb-sh%C3%ABrbimet-gjendja-aktuale-nga-Valdete-ZENUNI.pdf

Zenuni, V., & Rexha, B. (2007). Retrieved April 6, 2016, nga http://alpa.mali-it.eu/pub/aktet/vol/vol1/Aktet_Vol_I_Nr_2_pp_91_99.pdf

Zhang, S., Zhang, S., Chen, X., & Wu, S. (2010). Analysis and Research of Cloud Computing System Instance. *Second International Conference on Future Networks (ICFN).* Sanya, Hainan: IEEE.