Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Damir Arbula

RASPODIJELJENI ALGORITAM ZA ODREĐIVANJE POLOŽAJA ČVOROVA U NEUSIDRENOJ BEŽIČNOJ MREŽI OSJETILA

DOKTORSKI RAD

Zagreb, 2014.

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Damir Arbula

RASPODIJELJENI ALGORITAM ZA ODREĐIVANJE POLOŽAJA ČVOROVA U NEUSIDRENOJ BEŽIČNOJ MREŽI OSJETILA

DOKTORSKI RAD

Mentor:
Prof. dr. sc. Zdenko Kovačić

Zagreb, 2014.

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Damir Arbula

DISTRIBUTED ALGORITHM FOR NODE LOCALIZATION IN ANCHOR-FREE WIRELESS SENSOR NETWORK

DOCTORAL THESIS

Supervisor:
Professor Zdenko Kovačić, PhD

Zagreb, 2014

Doktorski rad izrađen je na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva, na Zavodu za automatiku i računalno inženjerstvo (ZARI).

Mentor: prof. dr. sc. Zdenko Kovačić

Doktorski rad ima: 161 stranicu.

Doktorski rad br.: _____

O mentoru

Zdenko Kovačić (1958) doktorirao je 1993. godine na Sveučilištu u Zagrebu. Redoviti je profesor u trajnom zvanju na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu i voditelj Laboratorija za robotiku i inteligentne sustave upravljanja. Kao dobitnik stipendije IREX-a za ak.g. 1990-1991., bio je gostujući istraživač u laboratoriju prof. Krishnana Ramua na Virginia Polytechnic Institute and State University u Blacksburgu, SAD. Dužnost predstojnika Zavoda za automatiku i računalno inženjerstvo obavljao je u razdoblju 2004/2008. Dobitnik je nagrade Sveučilišta u Zagrebu Fran Bošnjaković za 2013. godinu za istaknute rezultate, za unaprjeđenje i razvitak sustava automatskog upravljanja, fleksibilne automatizacije i robotike, zaprijenos znanja i stručni odgoj studenata i mladih stručnjaka te za javnu djelatnost i promicanje imena Fakulteta i Sveučilišta u zemlji i inozemstvu.

Bio je voditeljem više od 30 uspješno završenih razvojno-istraživačkih projekata. Objavio je 2 znanstvene monografije s međunarodnom recenzijom (izdavači Springer Verlag i Marcel Dekker (CRC Press)), 1 sveučilišni udžbenik, 4 poglavlja u knjigama s međunarodnom recenzijom, 9 radova u međunarodnim časopisima, 7 radova u domaćim časopisima te preko 130 radova u zbornicima skupova. Vodio je 53 završna rada, 103 diplomska rada (4 s naglaskom na znanstveno-istraživački rad), 7 magistarskih radova (1 nagrađen srebrnom plaketom FER-a „Josip Lončar"), 1 doktorsku disertaciju, te 10 studentskih radova (16 studenata), nagrađenih Rektorovom nagradom Sveučilišta u Zagrebu.

Od 1993. član je udruženja I.E.E.E. te je od 2010-2013. godine bio predsjednikom Odjela za robotiku i automatizaciju hrvatske sekcije IEEE-a. Predstavnik je Hrvatske u stručnim komisijama IFAC-a (Low Cost Automation, Adaptive Control, Robotics, Discrete Event Dynamic Systems) te IMEKO-a (Measurement in Robotics). Osnivatelj je i član Hrvatskog društva za robotiku (HDR) od 1994. godine. U razdoblju od 2006-2010 bio je predsjednik HDR-a, a trenutno je izabran za potpredsjednika i člana Izvršnog odbora. 2012. godine izabran je za prvog predsjednika Hrvatskog robotičkog saveza. Osnivač je i član Hrvatskog društva za komunikacije, računarstvo, elektroniku, mjerenja i automatiku KoREMA gdje je član predsjedništva društva te član uredništva časopisa Automatika.

Aktivno je sudjelovao u popularizaciji znanosti sudjelujući u različitim manifestacijama kao što su Festivali znanosti (Tehnički muzej, Zagreb), Festival tehničke kulture, Škole robotike u organizaciji HDR-a, demonstracijski seminari za srednješkolske profesore i drugo.

About the Supervisor

Zdenko Kovačić (1958) received his Ph.D.E.E. in 1993 from the University of Zagreb, Croatia. He is a full professor with tenure at the Faculty of Electrical Engineering and Computing, University of Zagreb and Head of the Laboratory for Robotics and Intelligent Control Systems. He spent 1990/91 as an IREX visiting researcher at the Bradley Department of Electrical Engineering, the Virginia Polytechnic Institute and State University, Blacksburg, USA. In 2004-2008 he was the Head of the Department of Control and Computer Engineering. He is the recipient of the University of Zagreb Award "Fran Bošnjakovic" for year 2013 for his distinctive scientific efforts in advancing and developing control, flexible automation and robotics systems with emphasis on robot control, cooperative robotic systems, intelligent control, service robotics, humanoids and walking robots.

He has been a principal investigator of more than 30 R&D projects. He is the author of 2 scientific monographs, 1 university textbook, 4 book chapters, 16 journal papers and more than 130 papers presented at the conferences. He was advisor of 1 Doctor of Science, 7 Masters of Science, 103 diploma engineers (4 with emphasis on R&D work), 53 bachelors of science, and 16 students who received University of Zagreb Rector Awards for their student works.

Since 1993 he is a member of IEEE. He served as the president of the Croatian IEEE Robotics and Automation Chapter in 2010-2013. He is a member of the IFAC Technical Committees for Robotics and for Adaptive Control and Tuning. He was the president of the Croatian Robotics Society 2005-2010 (also founder) and the vice-president (2011-2014). Since 2012 he is the elected president of the Croatian Robotic Association. He is also the founder and member of the presidency of KoREMA - Croatian Society for Communication, Computer, Electronics, Measurement and Control. He is a member of several journal editorial boards - Automatika (KoREMA), Journal of Intelligent Robotic Systems (Springer Verlag), Fire-fighting and management (Croatian Fire-fighting Community). He also served as a program chair, invited sessions chair, and track-chair on numerous international conferences.

He actively participated in the popularization of science by participating in various events such as the Festival of Science (Technical Museum, Zagreb), Festival of technical culture, Robotics Schools organized by Croatian Robotics Society, demonstration seminars for secondary school teachers and others.

Za Tihanu...
to my supergirl

Veliko hvala obitelji, prijateljima, kolegama i suradnicima za svu podršku, a posebno:

Prof.dr.sc. Zdenku Kovačiću – mom mentoru, za konstantnu, bezrezervnu podršku, motivaciju, stručno vodstvo, ali i prijateljske savjete i ugodnu radnu atmosferu.

Slavenu Glumcu – kolegi i suradniku, čovjeku bezgranične energije, talenta i zaraznog entuzijazma koji mi je pomogao u onim ključnim trenucima kada je trebalo povući.

Miroslavu Vrankiću i Kristijanu Lencu – prijateljima i kolegama, za podršku i razumijevanje, stučne i prijateljske savjete, te posebno za stavljanje tehničkog znanja u životni kontekst.

Sandiju Ljubiću i Goranu Mauši – prijateljima iz ureda, za smijeh, društvo i atmosferu zbog koje odlazak na posao čini veliko zadovoljstvo.

Prof. dr. sc. Vedranu Bilasu – članu povjerenstva za ocjenu i obranu doktorskog rada, za detaljno iščitavanje disertacije i mnogobrojne vrijedne savjete i korekcije.

Brojnim kolegama, djelatnicima Tehničkog fakulteta koji su mi pružali podršku, davali savjete, nesebično ustupali svu potrebnu opremu, ali i omogućavali ostanak na fakultetu i izvan radnog vremena.

Roditeljima i sestri – za bezuvjetnu podršku u svim mojim planovima i nezamjenjiv osjećaj da uvijek imam nekoga uz sebe.

Naposlijetku, veliko hvala mojim najsjajnijim zvijezdama Tihani i Jurju za svu ljubav i toplinu.

Sažetak

U području bežičnih mreža osjetila problem određivanja položaja čvorova predstavlja jedan od aktivnijih istraživačkih izazova. U ovom radu opisana je i analizirana izvorna metoda za procjenu kvalitete određivanja položaja čvorova korištenjem faktora geometrijskog slabljenja preciznosti dobivenog iz estimiranih lokacija. Isprojektiran je izvorni raspodijeljeni algoritam koji omogućava učinkovito korištenje energetskih, memorijskih i računalnih resursa, te precizno određivanje položaja čvorova na osnovu izmjerenih azimuta čvorova susjeda. Algoritam se zasniva na metodi spajanja lokalnih koordinatnih sustava kojom se omogućava veća točnost estimacije položaja, posebice za mreže s nekonveksnom topologijom. Metoda za procjenu kvalitete je potom iskorištena u heuristici koja, omogućava veću točnost algoritma za određivanje položaja, izbjegavanjem konfiguracija u kojima je povećana vjerojatnost pojave velikih pogrešaka u estimaciji. U cilju učinkovitog ispitivanja složenih strategija, metoda i heuristika, napisan je programski simulator raspodijeljenih algoritama u bežičnim mrežama osjetila. Dobiveni rezultati ispitivanja algoritma u simulatoru pokazuju smanjenu pogrešku estimacije položaja raspodijeljenog algoritma u odnosu na centraliziranu verziju u mrežama s većim prosječnim brojem susjeda kao i u mrežama s nekonveksnim toplogijama. Uvođenje navedene heuristike u mrežama s malim prosječnim brojem susjeda dodatno smanjuje estimacijsku pogrešku. Kako bi se verificirali rezultati dobiveni unutar programskog simulatora, algoritam je implementiran u ispitnoj mreži osjetila čiji čvorovi su opremljeni osjetilima za mjerenje azimuta. U tu svrhu je isprojektirano, izrađeno i ispitano osjetilo za uz pomoć kojeg se može estimirati azimut čvora susjeda, a koje se zasniva na mjerenju jakosti elektromagnetskog zračenja u infracrvenom dijelu spektra. Dodatno je izrađen i implementiran izvorni algoritam za estimaciju azimuta iz dobivenih mjerenja jakosti zračenja, te je obavljena kalibracija izrađenih osjetila.

Ključne riječi: bežične mreže osjetila, raspodijeljeni algoritmi, određivanje položaja

Extended Abstract

Distributed Algorithm for Node Localization in Anchor-free Wireless Sensor Network

This dissertation is the result of research in the field of distributed algorithms in wireless ad hoc networks, and specifically refers to the algorithms for nodes localization. Main research focus is the novel distributed algorithm for network nodes localization that is making use of carefully selected heuristic which allows the accurate localization by detecting and avoiding of node formations that have increased likelihood of large location estimation errors. Algorithm also implements methods for efficient use of energy, memory and computing resources. In order to effectively test different heuristics prior to implementation, a software framework for the simulation of distributed algorithms is developed. Based on the simulation results, algorithm is modified, implemented and verfied in a microcontroller network testbed. For this purpose, as a part of the verification, a novel angle of arrival sensor has been designed, manufactured and tested.

The first, introductory chapter summarizes the research field and research objectives. The second chapter "The problem of estimating the position of the nodes" describes the problem with an emphasis on measurement methods which form the basis of all localization algorithms. Also, in this chapter rigidity theory is presented. This theory gives the theoretical basis for determining the uniqueness of solutions. The third chapter, entitled "Review and analysis of related research" provides an overview of related research, formal classification of localization algorithms and the detailed description of the algorithms closely related to the proposed algorithm.

In the fourth chapter existing methods for assessing accuracy of the estimated locations are analyzed and mathematically described. These methods take into account relative node positions but neither one is suitable for usage in anchor free networks (i.e. networks without nodes with known positions). In the second part of the chapter, a novel method for assessing accuracy of estimated node locations in anchor free networks is presented and described. In the last part of the chapter proposed method has been implemented in the simulator and was used to estimate the quality of the location estimation in the initial network subclusters. Simulation results have been analyzed and a correlation between the estimated and true location estimation error has been quantified.

The fifth chapter presents novel distributed algorithm for localization of nodes in anchor free networks. The detailed description of each phase, its tasks, related problems, and methods for their solution is presented along with formal definition of used network formations. Subsequently, potential heuristics, based on the results obtained in the fourth chapter, are introduced and most appropriate is selected for implementation and

further assessment. Simulation results are given with the emphasis on the comparison of estimation accuracy, with and without the use of selected heuristic. In these simulations, selected heuristic was identified as the main source of accuracy improvement for the networks with low average number of neighbors per node. Finally, comparison with centralized version of the algorithm has been presented and it has shown that distributed algorithm can outperform corresponding centralized version, especially for the non-convex network topologies.

The sixth chapter entitled "Experimental verification" describes the microcontroller network testbed and details of the algorithm implementation and verification in this environment. For this purpose, a novel sensor for angle of arrival measurement is designed, manufactured and tested. This sensor uses infrared light detectors to estimate azimuth of the neighbouring nodes. Also, in order to reduce systematic errors, sensors are calibrated and novel azimuth estimation algorithm is implemented which manages to reduce standard deviation of estimation error below 1°. The results of the experiment are presented in the last part of the chapter with a focus on comparison with the results obtained in the simulator.

In the seventh chapter entitled "Pymote: distributed algorithm simulator" currently available network and distributed algorithm simulators are analyzed. This chapter gives the formal definition of the distributed computing environment. Its basic principles give the theoretical background on which the novel simulator was developed. Further discussion on platform selection, implementation details, and the ways in which the simulator can be upgraded with new features is given. In order to describe basic usage principles in which user can define, simulate, and analyze algorithms, using the interactive console or automated experiments, one sample algorithm implementation has been presented.

The final chapter comments the research, providing an overview, future work, and an outline of the research contributions achieved.

The research contributions of this dissertation thesis consist of: (i) the method for assessing the quality of the estimated node locations in anchor free networks based only on estimated locations (ii) novel distributed algorithm that accurately estimates the positions of nodes in a wireless sensor network using heuristic to detect and avoid inadequately localized network segments, especially for networks with non-convex topology (iii) distributed algorithms simulator for wireless sensor networks, which enables agile implementation and testing of complex strategies and methods and (iv) experimental verification of the novel distributed algorithm for node localization.

Keywords: wireless sensor networks, distributed algorithms, localization

Sadržaj

Popis slika

Popis tablica

Poglavlje 1

Uvod

Ubrzani razvoj tehnologija iz područja kao što su bežične komunikacije, elektronika, MEMS (engl. *microelectromechanical systems*) osjetila, te uređaja za prikupljanje energije iz okoliša doveo je do pojave jeftinih i energetski neovisnih uređaja malih dimenzija, koji mogu mjeriti, obrađivati i analizirati dobivene podatke te međusobno komunicirati na manjim udaljenostima. Takvi uređaji koje nazivamo bežični čvorovi (engl. *wireless nodes*) sastavni su dio bežičnih mreža osjetila.

Bežične mreže osjetila (engl. *wireless sensor networks – WSN*) u svom izvornom obliku spadaju u domenu takozvanih *ad-hoc* mreža kod kojih ne postoji prethodno definirana fiksna infrastruktura, već infrastrukturu čine sami uređaji koji komuniciraju. Teorijske osnove funkcioniranja bežičnih mreža osjetila razvijaju se unazad nekoliko desetljeća, a u zadnjih desetak godina zanimanje istraživačke zajednice dodatno je povećano zbog njihovog sve većeg potencijala u praktičnim primjenama.

WSN svoju, sve veću, primjenu nalaze u mnogim granama ljudske djelatnosti kao što su: nadziranje okoliša, biološka istraživanja, ekologija, promet, zaštita od požara i poplava, spašavanje, inteligentna poljoprivreda, nadgledanje i analiza ponašanja arhitektonskih struktura, industrijska mjerenja, nadzor infrastruktura kao što su energetske mreže i distribucija vode, inteligentne zgrade, određivanje lokacije i praćenje u mobilnoj robotici itd.

Usprkos mnogobrojnim primjenama bežičnih mreža osjetila, još uvijek postoji veliki broj istraživačkih izazova koji čekaju adekvatna rješenja kako bi se iskoristio puni potencijal koji nudi paradigma autonomne, sveprisutne i jeftine mreže mjernih čvorova koji omogućavaju detaljan uvid u nadzirane sustave i procese na način na koji to inače nebi bilo moguće ostvariti. Neki od otvorenih problema koji se aktivno istražuju su osiguranje kvalitete usluge, robusnost na promjene u topologiji, skalabilnost, upravljanje energijom, sigurnost, usmjeravanje prometa, sinkronizacija i određivanje položaja.

1.1 Određivanje položaja

Cilj postavljanja bežične mreže osjetila je u većini slučajeva vremenski kontinuiran, detaljan i sveobuhvatan uvid u nadzirane sustave i procese, a dobiveni podaci mogu rezultirati u konačnici donošenjem bržih i kvalitetnijih odluka. Kako bi navedeni uvid bio potpun, uz dobivene mjerne podatke, potreban je vremenski, ali i prostorni kontekst zbog kojeg čvorovi moraju poznavati svoju lokaciju. Bez podataka o točnoj lokaciji podaci često gube na značenju, no isto tako uz pogrešnu ili nedovoljno točnu lokaciju mogu biti i kontraproduktivni. Bežične mreže osjetila ostvaruju dodatan potencijal u primjenama u mrežama u kojima je dio čvorova mobilan te kada podaci o točnoj lokaciji imaju veliku težinu prilikom donošenja autonomnih odluka.

Pod pojmom "određivanje položaja" uključeno je određivanje lokacija i orijentacija čvorova. Ipak, kako se u mnogim primjenama, i posljedično metodama, traže i koriste isključivo lokacije, u radu će se u pojedinim dijelovima u ovisnosti o kontekstu, navoditi adekvatniji pojam.

Određivanje položaja je često tek jedan od problema koji, resursima i opremom ograničeni, čvorovi moraju riješiti. S obzirom na širok spektar vrlo različitih aplikacija uvjeti u kojima se taj problem rješava mogu biti vrlo specifični. Posljedica je veliki broj različitih pristupa istom problemu od kojih većina u određenom kontekstu ima komparativnu prednost nad drugima. Konačni izbor metode određivanja položaja u realnim sustavima uvjetuje odnos potreba i mogućnosti. Stoga, trenutni veliki izbor metoda nalaže sustavno ispitivanje svih, za pojedinu primjenu, relevantnih karakteristika koje će na kraju omogućiti objektivnu procjenu u pojedinoj situaciji.

Trenutno postoje mnoga otvorena pitanja u vezi odabira osnovnih kriterija vrednovanja, uvjeta u kojima algoritam funkcionira i načina na koji vrednovanje algoritama treba biti provedeno. Ipak naposlijetku, zaključak je kako algoritam mora optimalno iskoristiti sve prednosti sustava koji je na raspolaganju. To uključuje informacije do kojih može doći te načine na koje ih može obraditi u cilju umanjivanja ili onemogućavanja utjecaja inferiornijih karakteristika sustava na konačne rezultate. Upravo segment optimalnog iskorištenja svih dostupnih informacija predstavlja jednu od glavnih motivacija, hipoteza i doprinosa ovog rada.

1.2 Pregled rada

U radu se obrađuje izvorni raspodijeljeni algoritam za određivanje položaja, koji spada u domenu algoritama koji koriste tzv. *lokaliziraj-i-spoji* (engl. *Map & Stitch*) metodu [1, 2, 3, 4, 5]. Ti algoritmi se temelje na strategiji neovisnog određivanja položaja čvorova

manjih dijelova mreže te njihovim naknadnim spajanjem u veće, jedinstveno lokalizirane, cjeline. Spajanje dijelova mreže, odnosno koordinatnih sustava u kojima su definirane lokacije njihovih čvorova uključuje pronalazak optimalnih transformacijskih parametara (translacija, rotacija, skaliranje) na osnovi zajedničkih čvorova u oba sustava.

U dosadašnjem istraživanju u sklopu magistarskog rada [6], teorijski je definiran model spajanja dviju paralelno krutih formacija točaka. U skladu s tim dodatno su definirane posebne mrežne strukture odnosno logički dijelovi mreže temeljeni na topologiji i svojstvima krutosti. Te strukture u konačnici omogućavaju minimiziranje konačnog broja jedinstveno lokaliziranih dijelova mreže.

Nepouzdani i nepostojeći mjerni podaci, te nemogućnost obavljanja složenih izračuna razlog je što se u literaturi nailazi na veliki broj algoritama koji se u određenim fazama svog funkcioniranja oslanjaju na heuristike. Heuristika koja je analizirana u dosadašnjem istraživanju je utjecaj redoslijeda spajanja dijelova mreže na konačnu točnost estimiranih lokacija, a rezultat je pokazao kako nijedna od odabranih metoda po kojima se određivao redoslijed spajanja nije utjecala na konačnu kvalitetu. Bez obzira na takav zaključak, to istraživanje dalo je uvid kojim putem treba krenuti, a to je detekcija nekvalitetno lokaliziranih dijelova mreže i izbjegavanje njihovog korištenja u procesu spajanja. Taj smjer dodatno su potvrdili rezultati recentnih istraživanja [5, 7].

Lokaciju je često potrebno odrediti korištenjem nedostatnih ili nedovoljno pouzdanih mjerenja, što može uzrokovati velike pogreške. Odgovor na pitanje je li moguće uspješno odrediti položaje čvorova cijele mreže ili njenih pojedinih dijelova, daje teorija krutosti [8]. U dosadašnjem istraživanju fokus je stavljen na analizu tzv. paralelno krutih formacija koje proizlaze iz mjerenja azimuta među čvorovima susjedima.

Čak i kada je formacija kruta i kada se može garantirati da će estimirane lokacije čvorova biti jedinstvene, problem može predstavljati nepovoljna formacija čvorova kojima se određuje položaj, pri čemu se povećava vjerojatnost pojave velikih pogrešaka u estimaciji. To posebice dolazi do izražaja u fazi spajanja u kojoj pogreške u određivanju položaja čvorova jednog segmenta mogu propagirati i uzrokovati pogrešnu estimaciju položaja čvorova velikog dijela mreže. Prilikom određivanja položaja jednog čvora, koristeći mjerenja dobivena s više čvorova čija lokacija je poznata, kao što je to slučaj primjerice kod GPS-a, čvorovi mogu procijeniti kvalitetu određivanja položaja koristeći mjeru koja se naziva geometrijsko slabljenje preciznosti (engl. *Geometric dilution of precision - GDOP*) [9]. Jedan od doprinosa ove disertacije je izračun te mjere za slučaj kada se određuje relativan položaj čvorova u formaciji pri čemu nijedan od čvorova nema prethodnu informaciju o svom položaju. Korištenjem takvog relativnog GDOP-a u radu se pokazuje kako postoji mogućnost procjene kvalitete određivanja položaja koristeći upravo estimirane vrijednosti lokacija. To otvara mogućnost korištenja te mjere

u algoritmu za određivanje položaja. U nastavku istraživanja ispitan je jedan od načina na koji se može iskoristiti ova mjera koristeći izvorni raspodijeljeni algoritam za određivanje položaja uz primjenu heuristike kojom se učinkovito izbjegava korištenje podataka iz nedovoljno kvalitetno lokaliziranih dijelova mreže. Analizirane su i uspoređene kvalitete estimiranih lokacija centraliziranog algoritma, te raspodijeljenog algoritma sa i bez korištenja navedene heuristike, posebno za konveksne i nekonveksne topologije mreža.

Dodatni problem koji se proteže kroz dosadašnje istraživanje je adekvatno vrednovanje i usporedba različitih metoda za određivanje položaja. U raspoloživim istraživanjima kriteriji vrednovanja su brojni, a algoritmi su često analizirani i vrednovani samo u specifičnim uvjetima što vodi do potrebe za njihovom brzom implementacijom unutar simulacijskog okruženja. U tu svrhu u početnom dijelu istraživanja napisan je i programski simulator raspodijeljenih algoritama u bežičnim mrežama osjetila koji funkcionira na aplikacijskoj razini i omogućava implementaciju i brzo ispitivanje složenih strategija i metoda.

Naposlijetku, na temelju analize dosadašnjih istraživanja može se zaključiti kako, usprkos postojanju velikog broja metoda za određivanje položaja čvorova unutar mreža bez čvorova koji poznaju svoju lokaciju (tzv. sidra), relativno malo je onih koje su eksperimentalno ispitane na stvarnim mrežama. Posebice se to odnosi na područje algoritama koji se koriste mjerenjem azimuta među čvorovima u kojem u postojećoj literaturi nije pronađen nijedan takav primjer. Vjerojatni razlog tome je, još uvijek prisutan, nedostatak adekvatnih osjetila za tu namjenu. Najveći problem predstavljaju zahtjevi koji se postavljaju pred takvo osjetilo, pri tome se misli u prvom redu na dimenzije, a potom i na točnost mjerenja, te na domet. Ipak, i u tom vrlo specifičnom području postoje naznake napretka, pa se odnedavno može naići na informacije o razvoju komponenata koje bi se mogle primjeniti u tu svrhu [10]. S tim ciljem, u eksperimentalnom dijelu istraživanja je projektirano i izrađeno osjetilo za mjerenje azimuta te je provedeno umjeravanje i ispitivanje dometa i točnosti. U nastavku istraživanja tim osjetilima su opremljeni čvorovi bežične mreže osjetila, implementiran je i eksperimentalno verificiran raspodijeljeni algoritam za određivanje položaja čvorova.

U sklopu implementacije algoritma, a zbog ograničenih memorijskih i računalnih resursa izmjenjen je dio koji implementira analitičko rješenje korištenjem dekompozicije po singularnim vrijednostima (engl. *Singular Value Decomposition - SVD*). Klasa algoritama koja se pokazala kao kandidat za zamjenu i implementaciju, a koja ujedno zadovoljava i kriterije malih računalnih zahtjeva i mogućnost detekcije slabo lokaliziranih dijelova mreže su, između ostalih [11], algoritmi temeljeni na stohastičkom optimiranju, specifično mikroimunološki algoritmi [12].

U 2. poglavlju opisuje se problem određivanja položaja čvorova s naglaskom na mjerne

metode koje čine osnovu svih algoritama, teoriju krutosti koja daje teorijsku podlogu za određivanje jedinstvenosti rješenja, te vrednovanje dobivenih metoda. U 3. poglavlju je dan pregled algoritama srodnih izvornom algoritmu i to prema prethodno predstavljenim osnovnim klasifikacijama i opisu njihovih osnovnih dijelova. Slijedi poglavlje 4 u kojem je predstavljena izvorna metoda za procjenu kvalitete estimacije lokacija čvorova za neusidrene mreže koja se koristi u algoritmu za određivanje položaja, čiji detalji i rezultati simulacija su opisani u poglavlju 5, a čiji rezultati eksperimentalne verifikacije koja uključuje i projektiranje i izradu osjetila su predstavljeni u poglavlju 6. Ostatak rada čini opis simulatora u poglavlju 7 koji je razvijen i korišten u sklopu istraživanja.

Poglavlje 2

Problem određivanja položaja čvorova

Određivanje položaja čvorova u bežičnim mrežama osjetila je proces u kojem čvorovi stječu znanje o svojim koordinatama i orijentaciji unutar zadanog, apsolutnog koordinatnog sustava ili u odnosu na ostale čvorove unutar tzv. relativnog koordinatnog sustava. Usprkos jednostavnoj definiciji, rješavanje tog problema složen je zadatak kod kojeg veliki broj različitih parametara može igrati značajne uloge. Primjerice, opremljenost čvorova kao posljedica njihove željene veličine i cijene postavlja izravna ograničenja na metode mjerenja koje je moguće koristiti. Zatim okruženje, načini propagacije signala te općenito komunikacijski domet u sprezi s brojem i relativnim rasporedom čvorova određuju povezanost odnosno gustoću čvorova i topologiju mreže. Upravo gustoća čvorova u širem, odnosno topologija i pripadajući graf mreže u užem smislu, određuju je li moguće odrediti jedinstvene lokacije čvorova. U slučaju kada to nije moguće, potrebno je na razne načine pokušati izbjeći neadekvatan izbor rješenja koja mogu dovesti do propagacije pogrešaka i, u konačnici, pogrešno određenih lokacija većih segmenata mreže.

Kao što će se pokazati u ovom radu, postoji mnogo različitih pristupa problemu određivanja lokacije. Uzrok tome je veliki broj različitih primjena te, posljedično, karakteristika samih čvorova, mreža i okolina u kojima te mreže djeluju.

Metode određivanja lokacija čvorova mogu se u startu podijeliti na one koje se oslanjaju na infrastrukturu i/ili podatke izvan same mreže i potpuno autonomne metode. S obzirom na jedan od osnovnih koncepata samih bežičnih mreža osjetila, a to je autonomija, metode koje se oslanjaju na infrastrukturu neće biti posebno analizirane već samo spomenute. Primjer jedne od takvih metoda je korištenje GPS (engl. *Global Positioning System*) sustava. Osim konceptualnog nedostatka, s obzirom na potrošnju energije, veličinu i cijenu potrebne opreme u većini slučajeva, GPS ne predstavlja ni izvedbeno adekvatno rješenje. Također, GPS zahtjeva optičku vidljivost između prijamnika i dovoljnog

broja satelita što u mnogim slučajevima, kao što su zatvoreni prostori, nije moguće osigurati. Navedena ograničenja dovela su do intezivnog istraživanja alternativnih načina određivanja položaja, posebice onih koji se ne oslanjaju na fiksnu infrastrukturu. Pod pojmom određivanja lokacija čvorova u užem smislu smatra se autonomno određivanje lokacija od strane samih čvorova, isključivo korištenjem resursa mreže.

U ovom poglavlju opisuju se osnovni preduvjeti kako bi se uspješno odredile lokacije, poput mjerenja odnosa među čvorovima i teorije krutosti koja pruža odgovore je li moguće doći do jedinstvenog rješenja. Na kraju se daje kratak pregled metoda, uvjeta i kriterija vrednovanja metoda za određivanje položaja.

2.1 Mjerne metode

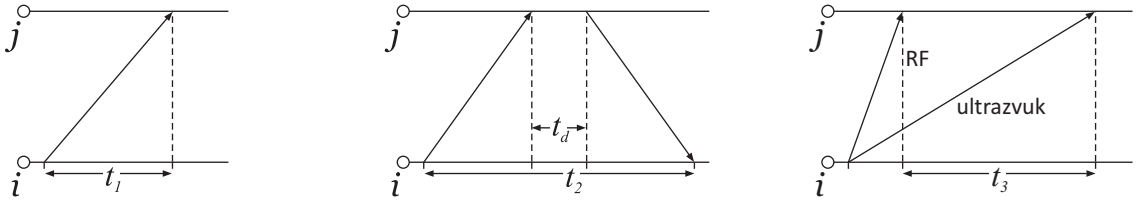
Preduvjet za uspješno određivanje položaja čvorova je mjerenje fizičkih odnosa među njima. Mjerenja se obavljaju različitim tehnikama koristeći poznate karakteristike propagacije radiofrekvencijskih, optičkih i akustičkih signala. Najčešće se mjeri jakost, vrijeme i smjer dolaska signala s ciljem određivanja odnosa među čvorovima kao što su *udaljenost* i *azimut*. Odabir tipa odnosa među čvorovima koji se mjeri, a ponekad i tipa mjerenja, izravno utječe na odabir algoritma za određivanje položaja čvorova.

2.1.1 Mjerenje udaljenosti

Udaljenost među čvorovima određuje se mjerenjem vremena dolaska signala (engl. *time of arrival – TOA*), mjerenja razlike u dolasku signala (engl. *time difference of arrival – TDOA*) ili mjerenjem jakosti signala (engl. *received signal strength – RSS*). Za određivanje udaljenosti iz dobivenih mjerenja u prva dva slučaja koristi se poznata brzina propagacije, a u trećem slučaju poznata atenuacija signala u mediju.

Vrijeme dolaska signala

Mjerenje vremena dolaska signala svodi se na određivanje vremena propagacije signala u jednom smjeru (engl. *one-way propagation time*), od čvora i do čvora j ili vremena obilaska (engl. *roundtrip time*) signala, od čvora i do čvora j i natrag. U prvom slučaju prikazanom na slici 2.1a početak i kraj intervala mjere različiti čvorovi, pa je preduvjet točnog mjerenja točna sinkronizacija lokalnih satova čvorova i i j. To je samo po sebi složen zadatak koji zahtjeva vrlo točne satove i sofisticirane mehanizme sinkronizacije, s obzirom na veliku brzinu i vrlo mala vremena propagacije radio signala. Pogodniji način mjerenja vremena propagacije je mjerenje obilaska kako je prikazano na slici 2.1b pri čemu se vrijeme mjeri u istom, polaznom čvoru, pa nije potrebna sinkronizacija. U



(a) jedan smjer (b) obilazak (c) razlika brzina

Slika 2.1: Mjerenje vremena propagacije.

tom slučaju nepoznanica je vrijeme t_d potrebno za prijem, obradu i ponovno odašiljanje signala od strane čvora j koje se mora oduzeti od ukupnog vremena. To vrijeme polazni čvor ili poznaje prethodnim umjeravanjem ili ga čvor j šalje u poruci. Metoda kojom se mjeri vrijeme propagacije u jednom smjeru, koja ne zahtjeva sinkronizaciju satova, prikazana je na slici 2.1c, a temelji se na razlici brzina propagacije radio signala i ultrazvučnog signala [13]. Simultanim slanjem radio i ultrazvučnog signala od strane čvora i odredišni čvor j može izmjeriti razliku vremena dolaska tih dvaju signala, a zatim poznavanjem brzina propagacije odrediti udaljenost. Ova metoda daje izuzetno točne rezultate, ali je uvjetovana opremljenošću čvorova ultrazvučnim odašiljačima i prijamnicima što povećava cijenu, veličinu i energetske zahtjeve samog čvora. Također postoji problem višestaznog širenja (engl. *multipath propagation*) ultrazvučnog signala, posebice u zatvorenim prostorima.

Razlika u vremenu dolaska signala

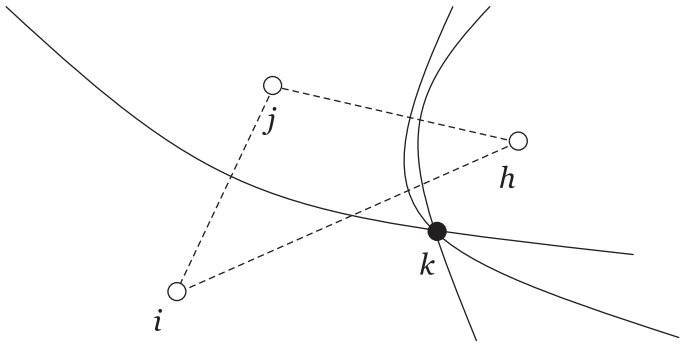
Metoda mjerenja razlike u vremenu dolaska signala omogućava izravno određivanje lokacije čvora k. Ako su čvorovi i i j vremenski sinkronizirani, i ako su odredili vremena t_i i t_j u kojima su primili signal s čvora k te ukoliko poznaju svoje lokacije tada vrijedi sljedeća jednakost:

$$\triangle t_{ij} \triangleq t_i - t_j = \frac{1}{c}(\|\boldsymbol{r}_i - \boldsymbol{r}_k\| - \|\boldsymbol{r}_j - \boldsymbol{r}_k\|), \quad i \neq j \tag{2.1}$$

gdje su $\boldsymbol{r}_i$, $\boldsymbol{r}_j$ i $\boldsymbol{r}_k$ lokacije čvorova i, j i k, a $\|\cdot\|$ označava euklidsku udaljenost. Za svaki par čvorova, jednadžba (2.1) opisuje po jednu hiperbolu, a lokacija čvora k nalazi se na njihovom sjecištu kako je prikazano na slici 2.2 [14].

Jakost primljenog signala

Odnos primljene snage $P_r(d)$ i odaslane snage signala P_t ovisan je o valnoj duljini λ, dobicima antena odašiljača i prijamnika G_t i G_r te o *udaljenosti prijamnika i odašiljača*



Slika 2.2: Određivanje lokacije na sjecištu hiperbola.

d prema sljedećoj formuli:

$$\frac{P_r(d)}{P_t} = \frac{G_t G_r \lambda^2}{(4\pi)^2 d^2}. \tag{2.2}$$

Jednadžba (2.2) vrijedi u slobodnom prostoru, međutim zbog vrlo složenog međudjelovanja okoline i signala kroz mnogobrojne refleksije, ogibe i raspršenja (engl. *non line of sight propagation – NLOS*), određivanje udaljenosti podložno je velikim pogreškama. Dodatni problem predstavlja povećanje mjerne pogreške sa udaljenošću zbog slabljenja signala odnosno zbog sve većeg utjecaja šuma. U [15] je uspoređena RSS metoda s TOA metodom te se pokazalo kako je pogreška mjerenja udaljenosti RSS metode veća za dva reda veličine. Pozitivne strane RSS metode su mali hardverski i energetski zahtjevi što je čini adekvatnom za primjenu u slučajevima u kojima se ne zahtijeva velika točnost estimiranih lokacija.

Mjerenje broja komunikacijskih skokova

U mrežama s velikim brojem čvorova moguće je, koristeći broj komunikacijskih skokova (engl. *hop-count*), dobiti predodžbu o relativnoj udaljenosti čvorova u mreži. Ukoliko je poznat i očekivani broj susjeda po čvoru n_s, tada se može izvesti i bolja procjena udaljenosti jednog komunikacijskog skoka [16]:

$$d_{\text{skok}} = R\left(1 + e^{-n_s} - \int_{-1}^{1} e^{-(n_s/\pi)\arccos t - t\sqrt{1-t^2}}\, dt\right). \tag{2.3}$$

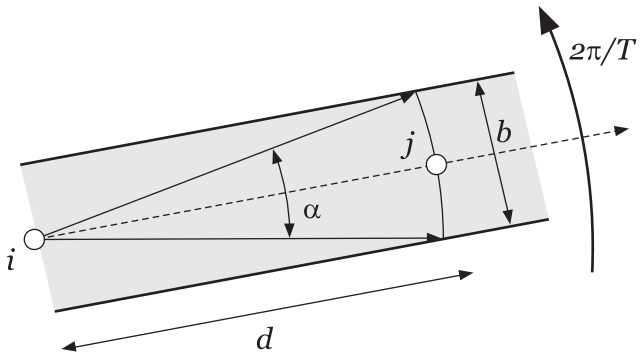
gdje je R maksimalni domet radio signala.

Aproksimacija duljine skoka (2.3) pokazala se korisnom za vrijednosti očekivanog broja susjeda po čvoru između 5 i 15. S obzirom da se svim čvorovima u dometu pridodjeljuje ista vrijednost duljine skoka prosječna pogreška iznose polovicu komunikacijskoo dometiraju, odnosno $R/2$. U mrežama s nehomogenom ili nepravilnom topologijom pretpostavka kako veliki broj skokova znači i veliku udaljenost nije točna te je pogreška pos-

ljedično još i veća. Kao i RSS, i ovu metodu mjerenja koristi se u slučajevima u kojima se ne zahtijeva velika točnost.

Metoda svjetionika

Mjerenje udaljenosti metodom svjetionika je primjer nestandardnih metoda mjerenja kojima se mjeri udaljenost, ali i izravno određuje lokacija čvorova. Ova metoda je predložena i opisana u [17] te pretpostavlja postojanje lokaliziranih *izvorišnih čvorova* i koji emitiraju rotirajuće svjetlosne zrake konstantnom kutnom brzinom ω prema slici 2.3.



Slika 2.3: Princip mjerenja udaljenosti uz pomoć metode svjetionika.

Odredišni čvorovi j, mjerenjem vremena t_1, u kojem detektiraju svjetlosnu zraku čvora i, te mjerenjem vremena T između dvije detekcije, mogu odrediti linearnu i kutnu brzinu zrake te izračunati polumjer, odnosno udaljenost među čvorovima koristeći sljedeću jednadžbu:

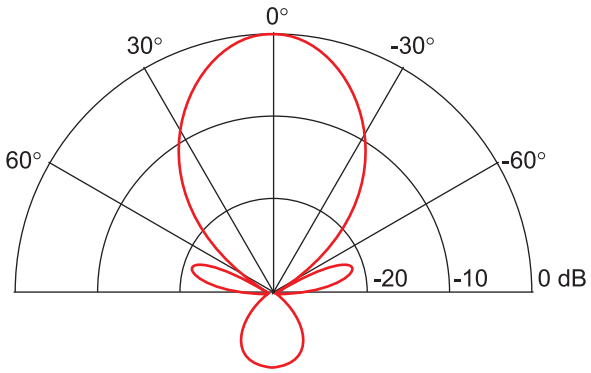
$$d \approx \frac{b}{2\sin(\alpha/2)} = \frac{b}{2\sin(\pi t_1/T)} \tag{2.4}$$

Praktični problem predstavlja zahtjev da zraka bude paralelna odnosno da ima konstantnu širinu b te da postoji optička vidljivost između izvorišnog i odredišnih čvorova. Prednost ovakvog pristupa je vrlo mala dimenzija i jednostavnost osjetila koji detektira zraku.

2.1.2 Mjerenje azimuta

Metode mjerenja kuta dolaska signala (engl. *Angle of Arrival – AoA*), odnosno azimuta susjednog čvora, mogu se podijeliti na dva pristupa: (1) mjerenje jakosti signala odnosno amplitudnog odziva te (2) mjerenje faznog odziva. Jedna od osnovnih metoda prvog pristupa zasniva se na dijagramu zračenja usmjerene antene, kojeg se može oblikovati elektronički ili samom konstrukcijom antene. Primjer dijagrama zračenja jedne usmjerene antene dan je na slici 2.4, a predstavlja grafički prikaz relativnog dobitka antene u

odnosu na smjer izvora signala. Fizičkim ili elektroničkim rotiranjem dijagrama prijamne antene, mijenja se dobitak G_r za signal koji dolazi iz određenog smjera, pa se prema jednadžbi (2.2), mijenja i jakost signala P_r. Kako prijamnik ne može razlikovati promjenu jakosti signala zbog rotacije od one koja nastaje zbog promjene snage odašiljanja često se uvodi još jedna antena s neusmjerenim dijagramom zračenja, koja služi za normalizaciju.



Slika 2.4: Tipični izgled dijagrama zračenja usmjerene antene.

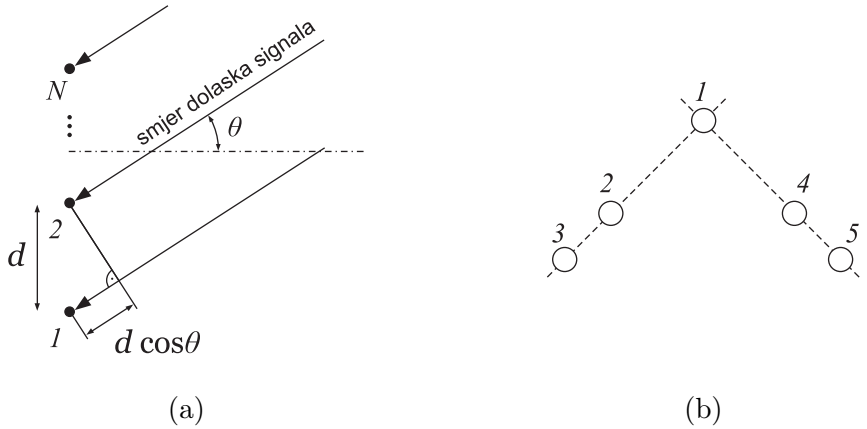
Veličina prijamne antene može biti mala u usporedbi s valnom duljinom signala što je velika prednost u odnosu na metodu mjerenja faznog odziva.

Metoda mjerenja faznog odziva uključuje korištenje niza neusmjerenih antena međusobno razmaknutih na udaljenost d reda veličine valne duljine signala što može biti neprihvatljivo s obzirom na uobičajeno male dimenzije čvorova. Jedan takav antenski niz od N antena prikazan je na slici 2.5a. Fazna razlika između susjednih antena, zbog razlike u duljini puta između pojedine antene i udaljenog odašiljača, ovisi o kutu dolaska signala, a može se prikazati jednadžbom

$$\varphi = 2\pi \frac{d \cos\theta}{\lambda} \tag{2.5}$$

prema kojoj je moguće odrediti i kut dolaska signala θ. Metoda slična opisanoj, a koja koristi fazne razlike ultrazvučnih signala obrađena je u [18]. Kako bi se riješila dvoznačnost preslikavanja fazne razlike u kut θ, u tom radu korišten je posebno oblikovan niz mikrofona prikazan na slici 2.5b, te su postignuti vrlo dobri rezultati – pogreška mjerenja kuta bila je u okviru 3°.

Mjerenje kuta općenito se temelji na propagaciji signala izravnom putanjom, te je vrlo važno razdvojiti utjecaj višestazne propagacije. Odašiljači u području milimetarskih valnih duljina mogu smanjiti veličinu potrebnog niza, a povećano gušenje u tom frekvencijskom području može pomoći ukloniti problem višestazne propagacije. Osim navedene dvije tehnike, postoje i posebna rješenja za određivanje azimuta, poput onog opisanog u [19] gdje je dio čvorova koji poznaje svoju lokaciju (sidra) opremljen sinkroni-



(a) (b)

Slika 2.5: Određivanje kuta iz fazne razlike primljenog signala. Na slici (a) prikazan je odnos između fazne razlike i kuta dolaska signala, a na slici (b) konfiguracija mikrofona kojom se rješava problem dvoznačnosti tog odnosa.

ziranim rotirajućim usmjerenim antenama, a obični čvorovi mjere razliku u vremenima u kojima detektiraju njihove pojedinačne signale.

2.1.3 Mjerenje profila signala

Mjerenje profila signala odnosi se na mjerenje jakosti, smjera ili vremena dolaska signala sa čvorova sidara. Preduvjet funkcioniranja metode je model prostora koji se sastoji od niza mjernih točaka s poznatom lokacijom u kojima se mjere profili, dok se profili na ostalim lokacijama dobivaju interpolacijom. Usporedbom izmjerenih vrijednosti u pojedinom čvoru, s vrijednostima u modelu prostora, može se estimirati lokacija čvora. Model je obično spremljen na centralnoj lokaciji na kojoj se odvija i navedena usporedba i estimacija. Pojedini model ovisi o položaju sidara, karakteristikama njihovih signala i okolini u kojoj signali propagiraju. Problem s ovom metodom je taj što će, ukoliko dođe do promjene u okolini, biti potrebno izraditi novi model. Kod pojedinih aplikacija predložena su razna inovativna rješenja tog problema. Primjerice, u [20] se predlaže korištenje mobilnih telefona u sprezi s mjerenjem broja koraka kao metodu za stvaranje i obnavljanje centraliziranog modela profila signala u zatvorenim prostorima.

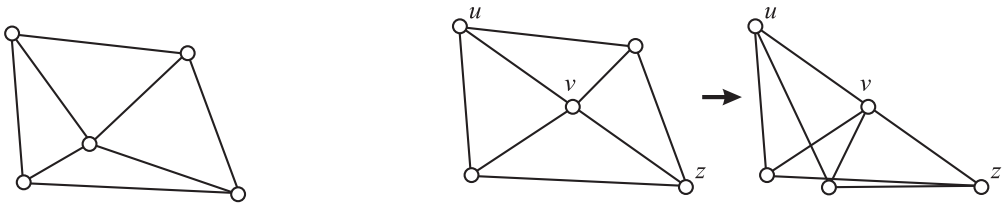
2.2 Teorija krutosti

Nakon što su obavljena mjerenja i određeni odnosi između pojedinih parova čvorova (udaljenost ili azimut, a u većem dijelu ovog poglavlja pretpostavlja se korištenje *udaljenosti*) potrebno je utvrditi jesu li dobiveni podaci dovoljni za određivanje jedinstvene lokacije svih čvorova mreže. Kako bi se moglo odgovoriti na to pitanje služimo se rezultatima iz područja teorije krutosti grafova (engl. *graph rigidity theory*). Mrežu pred-

stavljamo grafom $G = \{\mathcal{V},\mathcal{E}\}$, u kojem vrhovi $\mathcal{V} = \{1,2,\ldots,n\}$ predstavljaju čvorove, a bridovi $\mathcal{E}$ uređene parove čvorova (i,j) čija je međusobna udaljenost poznata. Udaljenost može biti dobivena mjerenjem ili izračunata, npr. ako je lokacija čvorova poznata (sidra) onda se njihova međusobna udaljenost (i azimut) može jednostavno izračunati.

Kako bi se graf mreže mogao reprezentirati u prostoru, definira se koncept formacije (engl. *formation, framework*). Formacija (G,p) je uređeni par koji se sastoji od grafa mreže G i funkcije p koja svakom vrhu pridružuje njegove koordinate u d-dimenzionalnom prostoru $p : \mathcal{V} \longmapsto \mathbb{R}^d$. Drugim riječima formacija predstavlja jednu od mogućih realizacija odgovarajućeg grafa mreže u d-dimenzionalnom prostoru. Dvije formacije (G,p) i (G,q) se smatraju *ekvivalentnima* ukoliko su njihovi bridovi iste duljine, odnosno $\|p(i) - p(j)\| = \|q(i) - q(j)\|$ za svaki par $(i,j) \in \mathcal{E}$. Dvije formacije (G,p) i (G,q) se smatraju *kongruentnima* (engl. *congruent*) ukoliko su udaljenost između *svih* vrhova jednake, odnosno $\|p(i) - p(j)\| = \|q(i) - q(j)\|$ za svaki par $i,j \in \mathcal{V}$. Ukoliko su sve ekvivalentne formacije određenog grafa ujedno i kongruentne tada je formacija *globalno kruta*, odnosno moguće je odrediti jedinstvene lokacije svih čvorova mreže. Pojam jedinstvene lokacije ovisi o kontekstu – ukoliko se radi o mreži u kojoj ne postoje sidra (neusidrena mreža) lokacija je jedinstvena do izometrije cijele mreže, odnosno transformacije koje čuva udaljenost među vrhovima poput translacije, rotacije i osne simetrije.

Određivanje je li pojedina formacija globalno kruta je NP-težak problem [8], no ako se problem ograniči samo na takozvane *generičke formacije* tada postaje ukrotiv (engl. *tractable*). Generička formacija je formacija u kojoj su koordinate vrhova algebarski nezavisne nad poljem racionalnih brojeva (slika 2.6a). Ukoliko formacija nije generička, moguć je slučaj u kojem su svi susjedi pojedinog vrha smješteni u $d-1$-dimenzionalnom prostoru. U tom slučaju jedna od mogućih ekvivalentnih, ali nekongurentnih, formacija uključuje onu u kojoj se navedeni vrh reflektira preko tog prostora odnosno ravine za $d = 3$ ili crte za $d = 2$ kao što je demonstrirano u primjeru na slici 2.6b.



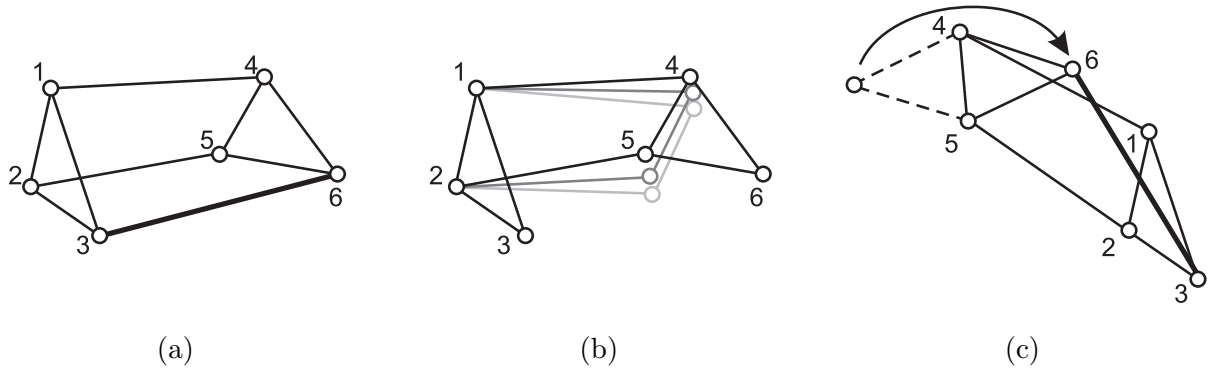
(a) (b)

Slika 2.6: Primjer realizacija (formacija) istog grafa pri čemu lokacije vrhova formacije (a) su generičke, (b) nisu generičke, jer su vrhovi u, v i z kolinearni te je moguće naći ekvivalentnu nekongruentnu formaciju.

Važno svojstvo generičkih formacija je da globalna krutost ovisi isključivo o grafu $\mathcal{G}$, odnosno ako je jedna generička realizacija pojedinog grafa globalno kruta onda su

sve realizacije tog grafa globalno krute. Ovo svojstvo omogućava donošenje zaključaka analizirajući samo graf odnosno samo pojedinu generičku realizaciju grafa kao primjerice u slučaju na slici 2.7. Zadan je kruti graf G koji nakon što mu oduzmemo jedan brid gubi to svojstvo, odnosno omogućeni su kontinuirani pregibi. Zahvaljujući tome, bez promjene duljina bridova, vrhove grafa se može dovesti u položaj u kojem je moguće "vratiti" oduzeti brid na način da mu duljina ostane ista, a da rezultirajuća formacija nije kongruentna početnoj. Grafovi koji su otporni na ovakve transformacije te ostaju kruti i nakon oduzimanja bilo kojeg brida se nazivaju *redundantno kruti* [21] i čine osnovu sljedećeg važnog teorema.

Teorem 1. *Dvodimenzionalne generičke formacije* (G,p) *su globalno krute ako i samo ako je graf* G *potpun graf s 2 ili 3 vrha* (K_2 *ili* K_3) *ili je 3-povezan i redundantno krut.*



(a) (b) (c)

Slika 2.7: Primjer krutog, ali ne i globalno krutog grafa (a), oduzimanjem brida (3,6) moguće je kontinuirano pregibanje četverokuta 1452 te nakon refleksije točke 6 (b), dobivena je konačna nekongruentna formacija koja zadovoljava isti skup ograničenja udaljenosti (c).

Teorem 1 potvrđuje intuitivan zaključak po kojem mreže koje imaju dovoljan broj i raspored mjerenja, odnosno čiji grafovi imaju dovoljan broj bridova adekvatno raspoređenih između svih vrhova, u pravilu imaju manji broj mogućih nekongruentnih realizacija, idealno samo jednu. Ukoliko je broj mjerenja, odnosno povezanost grafa dovoljno velika, to je dovoljan uvjet da graf bude krut što je pokazano sljedećim teoremom [22]:

Teorem 2. *Ako je graf* G *6-povezan onda je i globalno krut.*

Ako se prilikom određivanja položaja čvorova na vrijeme ne detektira mogućnost višestrukih rješenja, rezultat mogu biti velike pogreške u estimiranim lokacijama, ali i, što je još važnije, pogrešan topološki poredak čvorova odnosno orijentacija čvorova. Daljnjim korištenjem takvih lokacija omogućava se propagacija pogreške, a zbog pogrešne orijentacije cijeli segmenti mreže mogu biti preklopljeni, čak i ako se u daljnjem postupku više ne učine takve pogreške. Ova mogućnost je dodatno potencirana činjenicom da iznesena teorija ne uračunava realnu nesigurnost samih mjerenja, pa i u slučaju kada je u teoriji

situacija jasna, u praksi nije lako donijeti odluku. Unatoč tome, razvio se cijeli niz različitih pristupa problemu određivanja položaja koji upravo koriste rezultate teorije krutosti grafova u kombinaciji s poznavanjem karakteristika realnih mjerenja udaljenosti.

Jedan od pristupa izravno vezan za teoriju krutosti je korištenje kombinacije mjerenja, primjerice udaljenosti i azimuta. Primjerice u [23] definiraju se potrebni uvjeti za jedinstveno rješenje kada su između određenih čvorova poznate udaljenosti i/ili azimuti. Na temelju tih uvjeta predložena je metoda koja osigurava polinomsko vrijeme izvršavanja.

2.3 Vrednovanje algoritama

Veliki broj različitih aplikacija u kojima je potrebno obaviti lokalizaciju čvorova uzrok je isto tako velikom broju algoritama. Iako im je zadatak isti, svaka od tih metoda može pristupiti problemu lokalizacije sa svog jedinstvenog aspekta što rezultira različitim skupom kriterija koje pokušavaju zadovoljiti kao i različitim uvjetima u kojima se očekuje da te metode djeluju. Osim uvjeta u kojima djeluju i kriterija po kojima će se vrednovati pojedini algoritmi, vrlo bitno je definirati i odabrati jednu ili više referentnih metoda vrednovanja.

2.3.1 Metode vrednovanja

Algoritam se tijekom projektiranja ispituje na različite načine koji se mogu podijeliti u tri skupine od kojih svaka ima svoju ulogu u razvoju te usporedbi i odabiru algoritma.

Simulacije

U početnim fazama projektiranja algoritma u kojima se ispituju detalji funkcioniranja algoritma u potpuno kontroliranim uvjetima, koristeći pojednostavljene modele realnih procesa, koriste se simulacije. Uz pomoć simulacija mogu se otkriti konceptualni problemi u pristupu, ali također i donositi bitne odluke u daljnjim fazama razvoja. Primjerice, rezultati simulacija mogu pomoći procijeniti prednosti i mane centralizirane u odnosu na raspodijeljenu verziju algoritma što može biti dugoročno bitna odluka koju se ne može ili koju je iznimno teško učiniti u kasnijim fazama razvoja. Iako je dio detalja implementacije pojednostavljen, rezultati dobiveni simulacijama korisni su za usporedbu različitih pristupa po mnogim kriterijima. Također, simulacije su primjerene u pružanju odgovora na mnoga pitanja koja su nevezana izravno uz implementaciju poput primjerice rasporeda sidara u topologiji ili redoslijeda kojim se estimiraju lokacije čvorova.

Emulacije

Dodatni korak dalje prema ispitivanju sustava u realnim uvjetima čine emulacije. Pri tome se pojedini modeli korišteni u simulacijama zamjenjuju empirijskim podacima dobivenim u realnim mjerenjima. Primjerice mjerenja udaljenosti provedena realnim osjetilima često se koriste u emulacijama. Također, ono što emulacije razlikuje od simulacija je i izvedba algoritma koja često može biti na dovoljno niskoj razini za prevođenje (engl. *cross-compilation*) i implementaciju na samoj platformi.

Implementacija

Implementacija algoritma u realnim sustavima je daleko najteža, najskuplja i najmanje fleksibilna faza ispitivanja. S druge strane, to je najvažnija faza čiji krajnji cilj je eksperimentalna verifikacija funkcioniranja algoritma i vrednovanje u realnim uvjetima. U ovu fazu treba se ući vrlo oprezno s detaljnim planom i prethodno donešenim bitnim odlukama jer je za većinu promjena u implementacijskoj fazi potrebno puno vremena, ukoliko ih je uopće moguće izvesti. Primjerice, eksperimentiranje s različitim topologijama te prikupljanje podataka može biti vrlo otežano i podložno pogreškama. Zatim, ispitivanja u različitim okolinama mogu dati vrlo različite rezultate, pa je i prije same implementacije u ranijim fazama potrebno odrediti viziju upotrebe odnosno što konkretniju aplikaciju mreže te prema tome odrediti specifičnu težinu pojedinog kriterija vredovanja. Ukoliko se algoritam nije potvrdio i temeljito ispitao u simulacijama i emulacijama tada je u implementaciji često vrlo teško odrediti prave uzroke pojedinih problema: jesu li oni posljedica teorijskih ograničenja algoritma ili su posljedica konkretne implementacije odnosno realnih uvjeta u kojima funkcionira mreža.

2.3.2 Uvjeti vredovanja

Način na koji će se pojedini aspekti realnih sustava simulirati odnosno emulirati i razina detalja mogu imati veliki utjecaj na konačne rezultate. Upravo stoga je važno jasno definirati uvjete i odrediti kakav je njihov utjecaj na konačne rezultate kako bi analiza algoritama bila što kvalitetnija te kako bi se u daljnji proces moglo ući sa kvalitetnim odlukama.

Izbor ispitne topologije

Topologija mreže može imati izraziti utjecaj na rezultate algoritama lokalizacije. Postoji nekoliko tipičnih metoda koje možemo primijeniti za generiranje ispitnih topologija. Primjerice, izbor lokacija čvorova može biti ravnomjeran pri čemu su čvorovi postavljeni u pravilnu strukturu odnosno jednoliko razmaknuti jedni od drugih ili slučajan pri čemu

se čvorovima dodjeljuju nasumično odabrane lokacije. Topologije s ravnomjerno raspoređenim čvorovima se lakše vizualno analiziraju, dok slučajni raspored bolje odgovara realnom razmještaju čvorova kada se njihova lokacija ne može ručno podesiti.

Nadalje, na topologiju mreže u stvarnosti mogu utjecati prepreke, pa one mogu biti izrazito nepravilne. Pri tome fizička udaljenost jako odstupa od komunikacijske udaljenosti te algoritmi koji iskorištavaju heuristiku kojom se udaljenost procjenjuje brojem skokova mogu rezultirati velikim pogreškama. Primjeri takvih nepravilnih topologija koje se koriste u standardnim ispitivanjima su prstenaste topologije zatim nekonveksne topologije poput topologija u obliku slova U ili L. Algoritmi koji uspješno određuju položaj čvorova u nepravilnim topologijama sa slučajnim rasporedom čvorova su puno robusniji te bi, osim u iznimnim slučajevima, algoritme trebalo ispitivati koristeći takve topologije. Ukoliko algoritam koristi sidra, tada na njegove performanse utječe njihov broj i raspored, odnosno konveksnost formacije.

Model mjerenja

Rezultati mjerenja kojima se algoritmi za određivanje položaja služe podložni su pogreškama. Zato je važno odabrati model interpretacije rezultata mjerenja. Ukoliko nas zanima usporedba dvaju pristupa, pri čemu se analiza ne želi ograničiti na specifičnu opremu i okolinu ili čak tip mjerenja, tada bi jednostavniji modeli trebali biti dostatni. Jedan od takvih modela za mjerenje udaljenosti je model zašumljenog diska (engl. *noisy disk model*):

$$d_{ij} = \begin{cases} N(d_{ij},\sigma), & d_{ij} \leq d_{\max} \\ \text{nedefinirano}, & \text{inače} \end{cases} \tag{2.6}$$

u kojem je rezultat mjerenja slučajna varijabla s normalnom distribucijom oko prave vrijednosti i standardnom devijacijom σ.

Ukoliko je poznata metoda mjerenja, model je moguće dodatno prilagoditi primjerice za akustička mjerenja, ultra širokopojasna (engl. *ultra-wide band – UWB*) mjerenja ili za uvjete u kojima postoji mogućnost NLOS propagacije. Ponekad je potrebno učiniti korak više od modeliranja te u simulaciji koristiti empirijski dobivene podatke, primjerice kada je poznata mjerna oprema i okruženje u kojem se mreža nalazi i u kojem se obavljaju mjerenja.

Dodatni modeli

Ukoliko se osim potvrde valjanosti algoritma i razine točnosti simulacijom dodatno želi odrediti primjerice vrijeme izvršavanja ili procijeniti potrošnja energije, potrebno je modelirati odgovarajuće sustave. Kako mreža osjetila za komunikaciju koristi dijeljeni me-

dij, vrlo je bitno modelirati realan pristup tom mediju. Jedan od jednostavnijih predloženih načina modeliranja vjerojatnosti uspješnog prijenosa podataka je temeljen na modelu veze. Stanje veze je modelirano kao binarni Markovljev proces pri čemu ono može biti *dobro* ili *loše*. U ovisnosti u kojem je stanju veza, definirane su vjerojatnosti gubitka paketa.

Pogreške u komunikaciji dovode do povećanja vremena izvršavanja algoritma, ali i do povećanja potrošnje energije. Jedan od češće korištenih modela potrošnje energije koji uključuje i prijemnu i odašiljačku stranu komunikacijskog sustava može se izraziti sljedećom jednadžbom za ukupnu energiju potrebnu za prijenos k bitova na udaljenost d:

$$E(k,d) = kP_{Tx}T_B + P_t(d) + kP_{Rx}T_B \tag{2.7}$$

gdje je T_B vrijeme odašiljanja, P_t izračena snaga, a P_{Tx} i P_{Rx} snage potrebne za slanje odnosno prijam (demodulaciju i dekodiranje) jednog bita.

2.3.3 Kriteriji vrednovanja

Kako bi smo učinkovito vrednovali i usporedili različite metode i algoritme određivanja lokacija čvorova, potrebno je definirati kriterije po kojima se usporedba može izvršiti. Svaki od kriterija u konačnoj analizi ulazi s težinskim faktorom koji obično proizlazi iz konkretne primjene i uvjeta u kojima mreža treba funkcionirati. U ovom poglavlju formalno definiramo i opisujemo neke od najvažnijih i najkorištenijih kriterija vrednovanja te načine na koji se oni mjere i uspoređuju.

Točnost

Iz prethodnih poglavlja je jasno kako je odabir kriterija po kojima se provodi vrednovanje složen zadatak, te kako je s obzirom na to teško dva algoritma staviti u apsolutan odnos. Iako točnost nije jedini kriterij, a ne mora biti čak ni presudan, ipak glavni cilj algoritama za određivanje lokacija je *što točnije* odrediti lokaciju čvorova. Postoji mnogo načina na koje možemo vrednovati točnost estimiranih lokacija, a dijelimo ih po tome koriste li se točnim lokacijama čvorova ili ne.

U prvom slučaju računa se primjerice srednja apsolutna pogreška (engl. *mean absolute error – MAE*):

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} e_i}{n} \tag{2.8}$$

ili korijen srednje kvadratne pogreške (engl. *root mean square – RMS*):

$$\text{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{2.9}$$

pri čemu je e_i udaljenost između estimirane i točne lokacije čvora i.

Često se koristi i metrika (Frobenius) [24] koja uspoređuje stvarne udaljenosti između čvorova d_{ij} i estimirane $\hat{d}_{ij}$:

$$\text{FROB} = \sqrt{\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\hat{d}_{ij} - d_{ij})^2} \tag{2.10}$$

Na sličnoj metrici se zasniva i metoda koja ne koristi točne lokacije čvorova već uspoređuje estimirane udaljenosti s izmjerenima R_{ij}:

$$\text{pogreška} = \frac{2}{n(n-1)}\sum_{i,j,i;j} R_{ij} - e_i \tag{2.11}$$

Prednost ove metode je što se procjena pogreške estimiranih lokacija može učiniti u samoj mreži koja, dakako, ne poznaje točne lokacije čvorova. Ta činjenica se iskorištava u nekim algoritmima, primjerice kod metode simuliranog kaljenja. Također, ova metrika može biti prikladna kada se algoritam ispituje u fazi implementacije jer omogućava vrednovanje rezultata određivanja položaja bez potrebe za poznavanjem točnih lokacija čvorova na terenu.

Odabir metrike za evaluaciju točnosti lokalizacije može biti uvjetovan tipom pogreške koji je za odabranu metodu lokalizacije najizraženiji. Autori u [25] primjerice predlažu niz novih, do sada nekorištenih, metrika te ih analiziraju s obzirom na različite scenarije. U svakom od scenarija topologija estimiranih lokacija je rotirana, translatirana ili izobličena u odnosu na pravu. Cilj je odabrati metriku koja je najosjetljivija na karakterističnu tip pogreške za određeni scenario (algoritam, metodu) te na taj način omogućiti bolju analizu utjecaja različitih parametara na tu pogrešku.

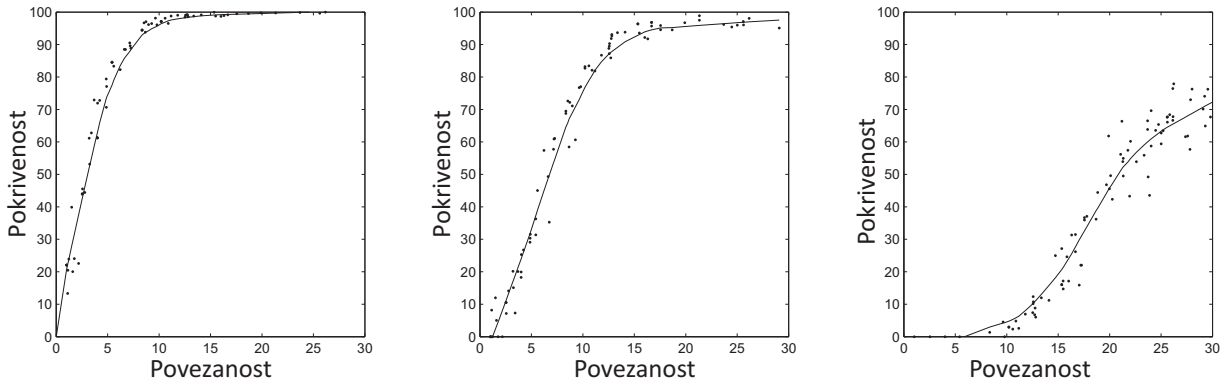
Skalabilnost

Jedan od primarnih ciljeva kod projektiranja algoritama je smanjivanje složenosti, odnosno trajanje izvođenja algoritma s obzirom na veličinu ulaznih parametara izraženo u O-notaciji. U slučaju algoritama za određivanje lokacija, ali i raspodijeljenih algoritama općenito, osnovni parametar po kojem se procjenjuje složenost algoritma je broj čvorova mreže. Osim računalne složenosti koja utječe na vrijeme izvršavanja klasičnih

algoritama, u bežičnim mrežama vrlo je bitna i komunikacijska složenost. Naime, čvorovi bežične mreže su energetski vrlo ograničenih mogućnosti, a energija potrebna za obradu podataka je nekoliko redova veličine manja od one potrebne za prijenos podataka između čvorova. Komunikacijska složenost se izražava obično u broju poruka, odnosno količini podataka koju je potrebno izmijeniti između čvorova. Pitanje i odluka koja je često ključna s obzirom na ovaj kriterij je izbor centralizirane odnosno raspodijeljene izvedbe algoritma. Pri tome raspodijeljene izvedbe imaju mnoge prednosti od kojih je najvažnija neovisnost o jednostavnoj topologiji u kojoj čvorovi šalju sve podatke u centralni čvor te ravnomjernija raspodjela računalnog i komunikacijskog opterećenja.

Pokrivenost

Često algoritmi za određivanje lokacija ne mogu odrediti lokacije svih čvorova mreže. Pokrivenost se mjeri kao udio broja čvorova za koje je moguće estimirati lokaciju u odnosu na ukupan broj čvorova u mreži. Razlozi zbog kojih čvorove nije moguće lokalizirati su njihov razmještaj i povezanost (prosječan broj susjeda po čvoru) te udio i razmještaj sidara. U kontekstu ispitivanja algoritama potrebno je analizirati koji su uzroci smanjene pokrivenosti vezani izravno uz metode koje se u njima koriste. Primarni uzrok zbog kojih je kod određenih algoritama pokrivenost ograničena je nemogućnost određivanja jedinstvene lokacije pojedinih čvorova. Posljedica korištenja estimirane lokacije s manjom vjerojatnošću jedinstvenog rješenja može dovesti do velike pogreške u lokaciji samog čvora, ali i većih segmenata mreže. Primjer algoritma u kojem se smanjuje pokrivenost kako bi se povećala točnost je AFL [1] kod kojega je uvjet da bi čvoru bila određena lokacija dovoljno visoka vjerojatnost da je dio grafa u kojem on leži krut. Na slici 2.8 vidimo kako za navedeni algoritam pokrivenost ovisi o standardnoj devijaciji pogreške mjerenja udaljenosti σ_d i o prosječnoj povezanosti čvorova mreže.



(a) $\sigma_d = 0$ cm (b) $\sigma_d = 1$ cm (c) $\sigma_d = 10$ cm

Slika 2.8: Pokrivenost za algoritam AFL.

Poglavlje 3

Pregled i analiza srodnih istraživanja

Cilj ovog poglavlja je predstaviti uže znanstveno područje unutar kojeg je obavljeno istraživanje. Kako bi to bilo moguće najprije su opisani najvažniji kriteriji prema kojima se klasificiraju algoritmi za određivanje položaja. Nakon toga su predstavljene osnovne metode određivanja položaja, a zatim je u tom kontekstu klasificiran izvorni algoritam koji čini okosnicu istraživanja. Naposlijetku, dan je pregled najvažnijih srodnih istraživanja s naglaskom na njihov odnos spram izvornog algoritma s ciljem određivanja relevantnosti metoda koje se koriste u algoritmu.

3.1 Klasifikacija algoritama

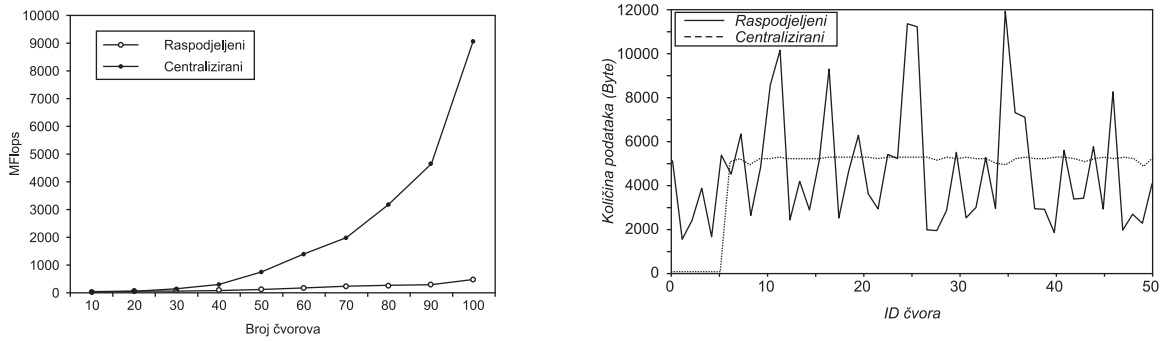
Algoritmi za određivanje položaja omogućavaju organizirano prikupljanje, izmjenu i obradu informacija u cilju određivanja položaja što većeg broja čvorova mreže. Informacije koje se koriste su estimirani odnosi (azimut, udaljenost) između čvorova te estimirane ili prave lokacije i orijentacije pojedinih čvorova. Bitna karakteristika dobro projektiranih algoritama je kvalitetno iskorištavanje dostupnih informacija. To podrazumijeva detekciju specifičnih slabosti i problema u pojedinoj mreži, kao što su slaba povezanost ili pogreške u mjerenjima, te njihovo izbjegavanje ili rješavanje korištenjem teorijske podloge poput teorije krutosti spomenute u prethodnom poglavlju. Kvalitetan algoritam također podrazumijeva optimalno korištenje ograničenih memorijskih, procesorskih i energetskih resursa, mogućnost rada u različitim uvjetima i robusnost s obzirom na topologiju mreže i broj čvorova.

Algoritmi se dijele prema različitim kriterijima, a jedan od najvažnijih je metoda obrade podataka koja može biti centralizirana ili raspodijeljena. Za centralizirane algoritme svi relevantni podaci se šalju u jedan, posebno odabran, centralni čvor koji ih zatim

obrađuje te estimira lokacije svih čvorova. Kod raspodijeljenih algoritama čvorovi sami obrađuju prikupljene podatke, što ravnomjernije i u idealnom slučaju istovremeno, te na taj način estimiraju svoju lokaciju. Prednost centraliziranih algoritama kod kojih centralni čvor posjeduje informacije iz cijele mreže je mogućnost implementacije proizvoljne strategije. Drugim riječima bilo koji raspodijeljeni algoritam se može učiniti centraliziranim, dok obrnuto ne vrijedi. Osnovni nedostatak centraliziranih rješenja u bežičnim mrežama je povećano i neravnomjerno raspoređeno komunikacijsko i računalno te posljedično energetsko opterećenje mreže. To posebice dolazi do izražaja u slučajevima u kojima se mreža sastoji od velikog broja čvorova. Tada centralni čvorovi moraju biti posebno opremljeni energetski i/ili računalnom snagom, što s druge strane smanjuje robusnost mreže jer funkcioniranje mreže ovisi o raspoloživosti jednog ili nekoliko posebnih čvorova.

S druge strane, mane raspodijeljenih algoritma se očituju u specifičnim uvjetima koje postavljaju. Osnovni uvjet je obrada podataka na lokalnoj razini. Drugim riječima, čvorovi mogu koristiti informacije, kao što su primjerice mjerenja, koja su dobili isključivo od svojih susjeda ili čvorova koji su od njih udaljeni za određeni, u pravilu mali, broj komunikacijskih skokova. Dodatni uvjet koji se postavlja pred raspodijeljene algoritme je paralelna, odnosno konkurentna, obrada podataka. Naime, komunikacijsko kašnjenje u bežičnim mrežama s više komunikacijskih skokova je često nepredvidivo te, kako bi rješenje bilo skalabilno, algoritam mora omogućiti čvorovima konkurentno izvršavanje što otežava njihovo projektiranje i verifikaciju.

Procjena je da se za bežični prijenos svakog bita koristi energija potrebna za 1000 do 2000 instrukcija [26] što prosljeđivanje poruka u mnogo skokova čini centralizirane algoritme energetski puno zahtjevnijima. Također prema [15, 27] komunikacijsko opterećenje je pravilnije raspoređeno po čvorovima koristeći raspodijeljenu inačicu algoritma kao što je prikazano na slici 3.1.



(a) (b)

Slika 3.1: Usporedba centralizirane i raspodijeljene inačice algoritma iz [27] s obzirom na (a) računalno opterećenje i (b) broj prenesenih bitova po čvoru.

Zbog činjenice da je obrada podataka većinom na lokalnoj razini i da se vrlo ograničena količina informacija prosljeđuje duž cijele mreže, u mrežama s puno čvorova, kao što su bežične mreže osjetila, puno učinkovitije je koristiti raspodijeljene algoritme.

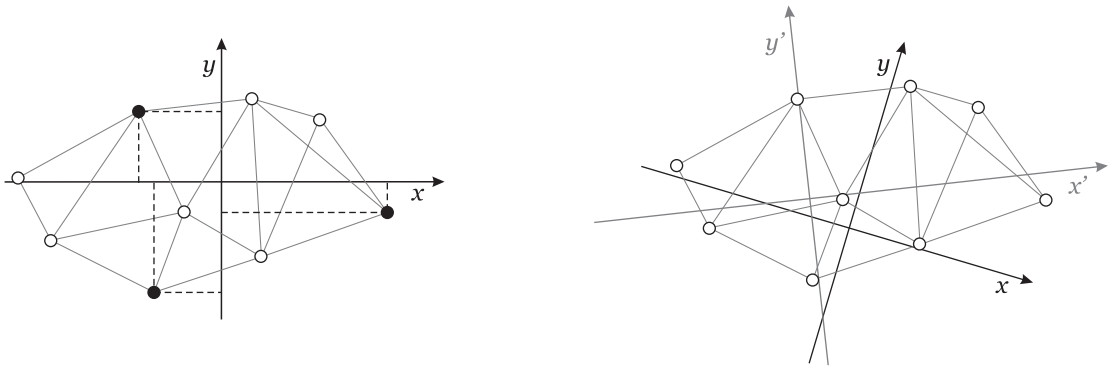
Izbor tipa algoritma ovisi i o drugim aspektima mreže na kojoj se obavlja implementacija. Primjerice, za mreže u kojima već postoji potreba za centraliziranim prikupljanjem i obradom podataka može biti praktičnije na isti način obaviti i određivanje položaja. Ipak, zaključno može se reći kako su, općenito gledajući, raspodijeljeni algoritmi robusniji, skalabilniji i energetski učinkovitiji, te ako su svi ostali parametri algoritma jednaki, ispravnije je odabrati *raspodijeljeno* rješenje.

Drugi tip klasifikacije je s obzirom na koordinatni sustav u kojem čvorovi estimiraju svoju lokaciju. Koordinatni sustav može biti definiran izvan mreže, pri čemu se naziva globalni odnosno apsolutni ili unutar same mreže te ga se u tom slučaju označava kao lokalni odnosno relativni.

Preduvjet određivanja lokacije u apsolutnom koordinatnom sustavu je postojanje određenog broja čvorova koji *a priori* poznaju svoju lokaciju u tom sustavu. Kao što je već spomenuto, ti čvorovi se nazivaju sidra (engl. *anchors*), a pripadajuće mreže se nazivaju usidrene mreže. Postojanje sidara je vrlo zahtjevan preduvjet s obzirom da čvorovi to znanje mogu steći jedino korištenjem infrastrukture izvan mreže što može biti skupo, nepraktično ili čak u nekim primjenama i neostvarivo.

S druge strane, u mnogim primjenama dovoljno je koristiti podatak o relativnoj lokaciji, primjerice za učinkovito usmjeravanje prometa unutar mreže. Za određivanje lokacija u relativnom koordinatnom sustavu odnosno u tzv. neusidrenim mrežama ne postoji preduvjet postojanja sidara, te je relativni koordinatni sustav definiran od strane čvorova u procesu određivanja lokacija.

Bitno je naglasiti kako se svaka metoda koja rješava problem određivanja položaja čvorova u neusidrenim mrežama može iskoristiti i za određivanje položaja u usidrenim mrežama dok obrnuto ne vrijedi. Naime metode koje su specifično namijenjene za korištenje usidrenim mrežama oslanjaju se na informacije koje dobivaju od sidara, pa čak i na njihov broj i relativan položaj unutar mreže, dok metode u neusidrenim mrežama te podatke, ukoliko su im dostupni, koriste samo za transformaciju položaja iz relativnog u apsolutni koordinatni sustav. Zaključno možemo reči, ukoliko su svi ostali parametri jednaki, ispravnije je koristiti generičko rješenje odnosno ono koje ima manje preduvjeta. U ovom slučaju to je algoritam za *neusidrene mreže*.



(a) Apsolutni koordinatni sustav. (b) Relativni koordinatni sustavi.

Slika 3.2: Primjer (a) usidrene mreže sa tri sidra koji poznaju svoju lokaciju unutar koordinatnog sustava definiranog izvan same mreže i (b) neusidrene mreže u kojoj se koordinatni sustav određuje od strane samih čvorova u mreži pri čemu koordinate čvorova u prikazanim primjerima koordinatnih sustava xy i $x'y'$ predstavljaju isto rješenje.

3.2 Osnovne metode za određivanje položaja čvorova

U nastavku je dan opis nekoliko najvažnijih pristupa u projektiranju algoritama za određivanje položaja. Te metode predstavljaju u nekim slučajevima gotova rješenja, a u drugim osnovne dijelove složenijih metoda čiji opis slijedi u dijelu 3.3.

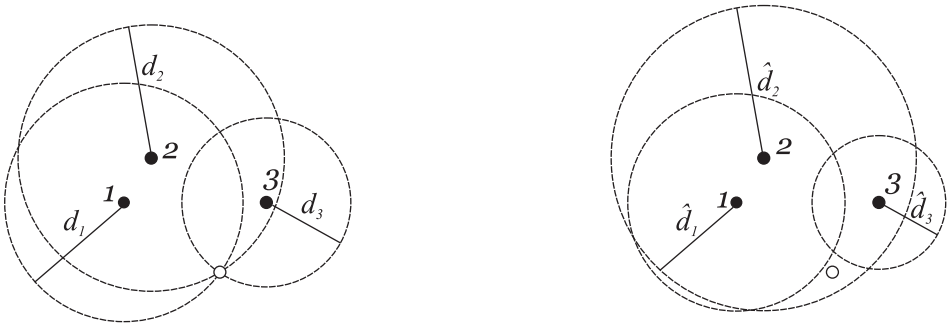
3.2.1 Trilateracija i triangulacija

Osnovna metoda određivanja lokacije pojedinog čvora koristi estimirane udaljenosti ili azimute prema nekoliko, u pravilu tri, sidra. Uz pretpostavku točnih mjerenja udaljenosti rješenje je trivijalno: čvor se nalazi na sjecištu kružnica s polumjerima jednakim udaljenostima i središtima u lokacijama čvorova kao što je prikazano na slici 3.3a. U praksi situacija se komplicira s obzirom na činjenicu da postoji pogreška mjerenja, pa rješenje više nije jednoznačno kao što se vidi na slici 3.3b.

Iz estimiranih udaljenosti lokacija se može odrediti koristeći metodu najmanjih kvadrata (engl. *least-squares estimation*) kao što je opisano u [15]. Ako se estimirane udaljenosti od ukupno k sidara $i \in [1,k]$ na lokaciji (x_i, y_i) označi s $\hat{d}_i$, možemo napisati jednadžbu:

$$\hat{d}_i - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} = 0 \tag{3.1}$$

pri čemu je (x_0,y_0) lokacija čvora koja se estimira.



(a) (b)

Slika 3.3: Trilateracija u slučaju kada su izmjerene udaljenosti od sidara (a) točne (b) netočne.

Nakon kvadriranja i sređivanja dobiva se:

$$-x_i^2 - y_i^2 = (x_0^2 + y_0^2) + x_0(-2x_i) + y_0(-2y_i) - \hat{d}_i^2 \tag{3.2}$$

Ukoliko jednadžbu (3.2) raspišemo za svako sidro te, kako bi se riješili elementa $(x_0^2 + y_0^2)$, od prvih $k-1$ jednadžbi oduzmemo k-tu i zatim ispišemo u matričnom obliku, dobivamo jednadžbu $y = bX$ pri čemu je:

$$y = \begin{bmatrix} x_k^2 - x_1^2 + y_k^2 - y_1^2 \\ x_k^2 - x_2^2 + y_k^2 - y_2^2 \\ \vdots \\ x_k^2 - x_{k-1}^2 + y_k^2 - y_{k-1}^2 \end{bmatrix} \quad X = \begin{bmatrix} 2(x_k - x_1) & 2(y_k - y_1) & \hat{d}_k^2 - \hat{d}_1^2 \\ 2(x_k - x_2) & 2(y_k - y_2) & \hat{d}_k^2 - \hat{d}_2^2 \\ \vdots & \vdots & \vdots \\ 2(x_k - x_{k-1}) & 2(y_k - y_{k-1}) & \hat{d}_k^2 - \hat{d}_{k-1}^2 \end{bmatrix} \quad b = \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{3.3}$$

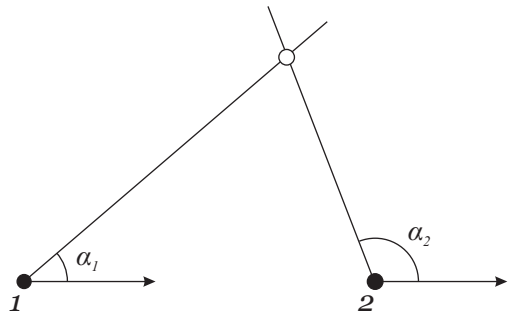
Rješenje ovog sustava, odnosno estimirana lokacija jednaka je $b = (X^T X)^{-1} X^T y$.

Ako se umjesto udaljenosti mjere azimuti od sidara prema čvoru, kao što je prikazano na slici 3.4, tada se geometrijski princip estimacije lokacije naziva triangulacija, a svodi se na pronalazak sjecišta barem dva polupravca koji započinju u sidrima i prostiru se u smjeru izmjerenog azimuta.

Analitičko rješenje za sjecište dano je jednadžbama:

$$\begin{aligned} x_0 &= x_2 + \cos(\alpha_2)\frac{y_2 - y_1 - \tan(\alpha_1)(x_2 - x_1)}{\cos(\alpha_2)\tan(\alpha_1) - \sin(\alpha_2)} \\ y_0 &= y_2 + \sin(\alpha_2)\frac{y_2 - y_1 - \tan(\alpha_1)(x_2 - x_1)}{\cos(\alpha_2)\tan(\alpha_1) - \sin(\alpha_2)} \end{aligned} \tag{3.4}$$

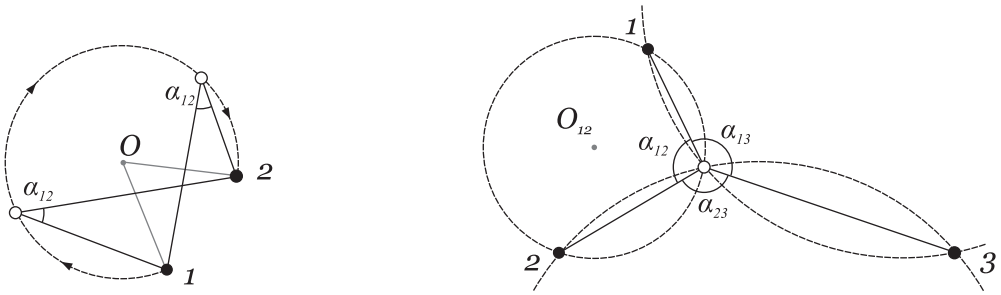
Preduvjet jedinstvenog rješenja za slučaj s dva sidra je poznavanje globalne orijentacije. Drugim riječima, sidra moraju biti opremljena kompasima. Kada taj uvjet nije



Slika 3.4: Triangulacija u slučaju kada su azimuti izmjereni u odnosu na poznatu referentnu orijentaciju.

zadovoljen, potrebna su barem tri sidra te i u tom slučaju postoji izravno analitičko rješenje dano u [28].

Princip triangulacije možemo svesti na trilateraciju kao što je opisano u [29], a osnovna ideja je da se za svaki par sidara iz njihovih lokacija i razlike u azimutima odredi kružnica na kojoj se čvor mora nalaziti (slika 3.5a). Za n izmjerenih azimuta dobiva se $\binom{n}{2}$ kružnica s poznatom lokacijom središta i polumjerom te se lokacija čvora može estimirati na isti način kao i za problem trilateracije, što je prikazano na slici 3.5b.



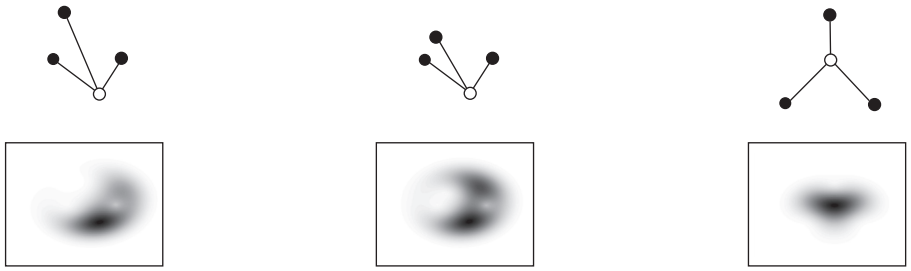
(a) (b)

Slika 3.5: (a) Moguća lokacija čvora uz izmjerenu razliku azimuta prema paru sidara je na kružnici definiranoj lokacijama sidara i izmjerenom razlikom azimuta. (b) Problem triangulacije definiran kao problem trilateracije, estimirana lokacija čvora je na presjecištu kružnica.

S obzirom na jednostavnost i izravnost, trilateracija i triangulacija čine osnovu mnogih metoda od kojih će pojedine biti spomenuti u pregledu srodnih istraživanja. Jedan od ključnih zadataka tih algoritama je procijeniti kvalitetu estimirane lokacije i držati pod kontrolom pogrešku u određivanju lokacije odnosno moguću propagaciju iste.

Jedna od metoda kojom se estimira kvaliteta trilateracije opisana je u [30], a svodi na određivanje vjerojatnosti da se lokacija čvora nalazi unutar kružnice s obzirom na raspored sidara, estimiranu lokaciju i varijancu mjerenja udaljenosti. Na slici 3.6 grafički je prikazana raspodjela vjerojatnosti lokacije čvora za različite formacije sidara.

Za triangulaciju, slično kao i u slučaju trilateracije, ukoliko mjerenja nisu dovoljno



(a) (b) (c)

Slika 3.6: Kvaliteta trilateracije izražena kao gustoća vjerojatnosti lokacije čvora s obzirom na formaciju sidara i mjernu pogrešku koja je u sva tri prikazana slučaja ista.

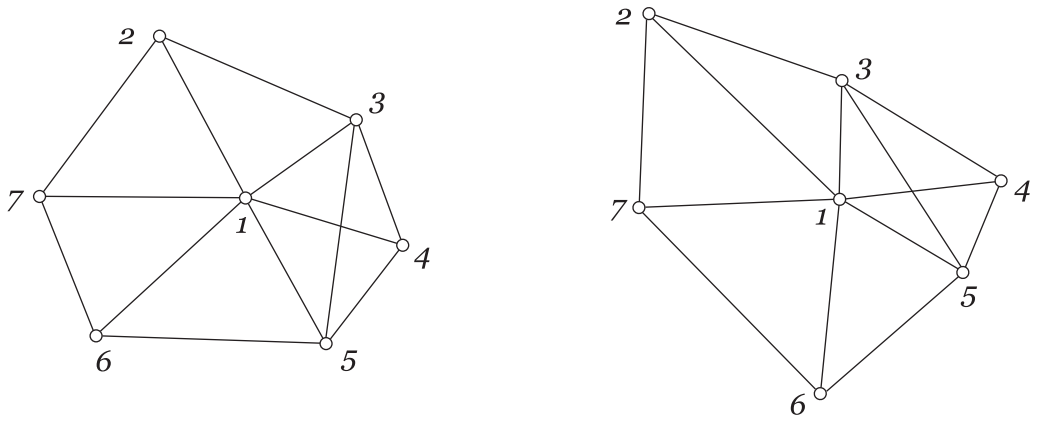
točna postoji potreba za analizom i kontrolom pogreške. U jednom od recentnih radova [31] autori opisuju analitičko rješenje zasnovano na metodi najmanjih kvadrata te, nakon analize pogrešaka, izvode inačice osnovnog rješenja u kojima ih kompenziraju. Kao i kod estimacije kvalitete trilateracije, jedna od informacija koja se koristi su varijance mjerenja azimuta. Iako je u radu predstavljena i inačica koja ih ne koristi, inačice koje ih koriste postižu najbolje rezultate.

3.2.2 Višedimenzionalno skaliranje

Za razliku od prethodno opisanih metoda trilateracije i triangulacije kojima se određuje lokacija jednog čvora koristeći mjerenja s više sidara, u nastavku se opisuje metoda koja funkcionira u neusidrenim mrežama i kojom se istovremeno određuju lokacije više čvorova s obzirom na opcionalno nepotpunu matricu izmjerenih odnosa među njima. Višedimenzionalno skaliranje (engl. *multidimensional scaling – MDS*) [32] je metoda kojom se analizira sličnost između pojedinih objekata. Objekti su inicijalno pozicionirani u višedimenzionalnom prostoru, pri čemu je međusobna udaljenost između dva objekta mjera njihove sličnosti. Cilj MDS metode je predstaviti odnosno vizualizirati iste objekte u prostoru s manje dimenzija (tipično 1, 2 ili 3-dimenzionalnom) te pri tome čim više zadržati početne odnose. Kod algoritama za određivanje položaja, objekti su čvorovi, a odnosi među objektima mogu biti bilo koji od opisanih: udaljenost, azimut, susjedstvo.

Matematičke metode kojima se provodi višedimenzionalno skaliranje uključuju transformacije matrice odnosa među objektima kao što je primjerice rastav singularnih vrijednosti (engl. *singular value decomposition – SVD*). Zbog toga što matrica treba biti potpuna, ukoliko između pojedinih čvorova nije moguće izvesti mjerenje, tada je njihov međusobni odnos potrebno estimirati na osnovu postojećih mjerenja što dovodi do pogrešaka čak i u relativno dobro povezanim i malim mrežama (slika 3.7).

Rezultati dobiveni uz pomoć MDS-a mogu biti dodatno optimirani koristeći metode najveće vjerodostojnosti (engl. *maximum likelihood - ML*), međutim to su u većini slu-



(a) (b)

Slika 3.7: (a) Prave lokacije čvorova. Između pojedinih čvorova ne postoji mogućnost mjerenja udaljenosti, pa se koristi estimirana udaljenost. (b) Estimirane lokacije čvorova dobivene višedimenzionalnim skaliranjem pri čemu dolazi do pogrešaka zbog estimiranih (precijenjenih) udaljenosti.

jeva iterativne centralizirane metode i kao takve nisu adekvatne za primjenu u bežičnim mrežama osjetila.

3.2.3 Linearno programiranje

Problem lokalizacije matematički se može formulirati kao problem iz domene konveksne optimizacije koji se rješava metodama linearnog i semidefinitnog programiranja (engl. *semidefinite programming – SDP*). Problem je definiran na sljedeći način:

$$\begin{aligned} &\min && \boldsymbol{c}^T\boldsymbol{X} \\ &\text{uz ograničenja} && F(\boldsymbol{X}) \geq 0 \end{aligned} \tag{3.5}$$

pri čemu je $\boldsymbol{X} = [\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots \boldsymbol{X}_n]^T$, a $\boldsymbol{X}_i = [x_i, y_i]^T$ su koordinate čvorova. Ograničenja se definiraju primjerice kao $\|\boldsymbol{X}_i - \boldsymbol{X}_j\| \leq R$ odnosno koordinate čvorova susjeda mogu biti udaljene najviše za komunikacijski domet R. Funkcija cilja $\boldsymbol{c}^T\boldsymbol{X}$ se može zanemariti ili se može postaviti na -1 odnosno 1 za određeni čvor i, a za sve druge na 0 te se time rješenje problema lokalizacije za taj čvor ograničava na pravokutni segment ravnine omeđen minimalnom odnosno maksimalnom vrijednosti za x_i odnosno y_i koordinatu.

Ova metoda daje zadovoljavajuće rezultate ukoliko su sidra postavljena u konveksnu formaciju na vanjske rubove mreže. U [33] autori dodaju nova ograničenja poput onih u kojima čvorovima nije dozvoljeno biti na manjoj udaljenosti ili onih u kojima čvorovi moraju biti točno na određenim udaljenostima. Problem s metodama optimiranja je da su računalno prilično zahtjevne i centralizirane, pa je zbog toga i komunikacijsko

opterećenje pojedinih čvorova veliko. Također, ove metode ovise o povezanosti i slabo se nose s vrlo izglednim situacijama u kojima postoji više mogućih rješenja i posljedicama odabira pogrešnih rješenja.

3.2.4 Vektor udaljenosti

DV-hop [34] algoritam, u čijem imenu DV označava vektor udaljenosti (engl. *distance vector*) funkcionira isključivo u usidrenim mrežama. Algoritam započinje tako da sva sidra šalju svim čvorovima u dometu svoje koordinate. Koordinate se prosljeđuju od čvora do čvora duž cijele mreže, a svaki čvor, uz prosljeđivanje, sprema primljene koordinate sidara i za svako sidro njegovu najmanju udaljenost u komunikacijskim skokovima. Naime, čvor od istog sidra može primiti podatke duž više različitih putova. Nakon toga, iz tih podataka svako sidro i koje također održava informacije o koordinatama i udaljenostima h_j od drugih sidara j određuje prosječnu duljinu skoka te taj podatak šalje u mrežu:

$$c_i = \frac{\sum_{i \neq j} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum_{i \neq j} h_j}. \tag{3.6}$$

Naposljetku svi čvorovi koji imaju podatke o prosječnoj duljini skoka i udaljenosti u skokovima od barem tri sidra mogu izračunati svoje koordinate.

DV-*distance* je modificirana inačica DV-hop algoritma koja koristi izmjerene udaljenosti među čvorovima. Algoritam je vrlo jednostavan te, uz dobar raspored sidara, konveksnu topologiju i homogenu raspodjelu čvorova rezultira zadovoljavajućom točnosti.

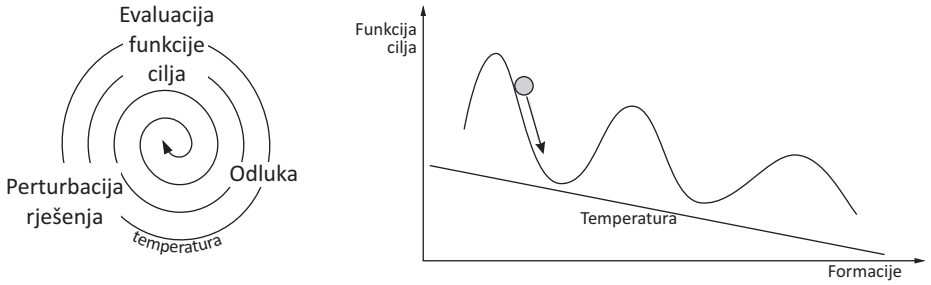
3.2.5 Stohastičko optimiranje

Pristup rješavanju problema lokalizacije metodama stohastičkog optimiranja sastoji se od formulacije jednog dijela problema optimizacije koristeći slučajne varijable. Taj dio mogu biti ograničenja ili funkcija cilja, a u problemu lokalizacije to su iteracije potencijalnih (predloženih) rješenja, odnosno koordinate čvorova. U potrazi za pravim rješenjem iz skupa takvih slučajnih rješenja koriste se razne tehnike kombinatoričkog optimiranja. Jedna od popularnijih metoda za problem lokalizacije je simulirano kaljenje (engl. *simulated annealing – SA*) [35]. Tehnika se temelji na oponašanju fizikalnog procesa hlađenja i formacije kristala pri čemu polagano i kontrolirano hlađenje omogućava postizanje nižih energetskih stanja i, posljedično, pravilniju strukturu kristala. U problemu optimizacije niža energetska stanja znače manju vrijednost funkcije cilja, odnosno funkcije koju želimo minimizirati. Drugim riječima, algoritam simulira polagano snižavanje temperature te usporedo s tim iterativno obavlja određeni broj perturbacija rješenja (koordinata) koje su po iznosu proporcionalne trenutnoj temperaturi, a po smjeru slučajne. Za svako novo

rješenje ponovno se evaluira funkcija cilja. Jedan primjer funkcije cilja je:

$$\sum_{i=1}^{n} \sum_{j \in N_i} (\hat{d}_{ij} - d_{ij})^2 \tag{3.7}$$

gdje je d_{ij} izmjerena udaljenost između susjednih čvorova i i j, a $\hat{d}_{ij}$ udaljenost izračunata iz samog rješenja koje se evaluira. Ukoliko je dobiveni iznos funkcije cilja manji od prethodnog novo rješenje se prihvaća, a ukoliko nije određuje se vjerojatnost prihvaćanja koja je proporcionalna trenutnoj temperaturi. Na taj način se omogućava algoritmu izlazak iz lokalnih minimuma. Algoritam završava ili kad funkcija cilja postane manja od određenog definiranog praga ili kada se dosegne konačna, također predefinirana temperatura.



Slika 3.8: Simulirano kaljenje.

Samom metodom simuliranog kaljenja se ne rješava problem višestrukih rješenja koji se javlja kada graf mreže nije krut i/ili su mjerenja netočna, a koji može dovesti do velikih pogrešaka. Zbog toga se često koriste dodatne metode u kojima se na razne načine pokušava umanjiti efekt tog problema. Primjerice u [35] se koristi pretpostavka da pogrešno lokalizirani čvorovi završavaju na lokacijama koje su u komunikacijskom dometu čvorova s kojima nisu u susjedstvu. To je indikacija moguće pogreške te se taj podatak uvodi u funkciju cilja druge faze koja koristi čvorove kojima nije indicirana takva pogreška kao nova sidra. Algoritmi koji se zasnivaju na simuliranom kaljenju pokazali su se kao robusni i uspješni u izbjegavanju lokalnih minimuma. No to vrijedi samo za slučajeve kada je povezanost čvorova velika odnosno kada je metoda indikacija pogreške zbog višestrukih rješenja učinkovita. Također, s obzirom na velike računalne zahtjeve, u literaturi su obrađene samo centralizirane izvedbe.

3.2.6 Ostale metode

Osim navedenih metoda postoji još niz različitih pristupa od kojih valja istaknuti sljedeće.

Metode temeljene na jakosti primljenog signala

Ova klasa metoda je vrlo atraktivna s aspekta implementacije s obzirom da većina čvorova ima mogućnost mjerenja jakoste primljenog signala. Iako je pristupa puno, dijele se na one koji koriste veliki broj mjerenja jakosti signala svih čvorova te na one koji se oslanjaju na mjerenja jakosti signala isključivo od čvorova koji znaju svoju lokaciju.

Dodatno, od čvorova koji poznaju svoju lokaciju može se napraviti selekcija te koristiti samo oni koji su zadovoljili određene uvjete. Primjerice u [36] autori koriste nekoliko kriterija za selekciju koji u pripremnoj fazi preferiraju čvorove s boljom korelacijom između udaljenosti i jakosti primljenog signala. Korelacija ovisi i o samoj udaljenosti, pa se u drugoj fazi za izabrane čvorove računaju propagacijski parametri odnosno pogreška u odnosu na udaljenost. Tako se u zadnjoj fazi prilikom same estimacije lokacija izbjegava korištenje čvorova čija jakost signala je izvan područja s dovoljno dobrom korelacijom između mjerenja i udaljenosti. Zbog činjenice da je jakost primljenog signala, uz udaljenost, ovisna i o drugim parametrima od kojih mnogi ovise o promjenjivoj okolini, točnost ovakvih metoda je mala.

Metode temeljene na strojnom učenju

Ove metode služe kako bi se iz velike količine izmjerenih podataka, a u većini slučajeva to su jakost primljenih signala, mogle odrediti lokacije čvorova. Dijele se s obzirom na način na koji je definiran problem lokalizacije: (i) kao problem klasifikacije ili (ii) kao problem regresije. U prvom slučaju prostor je podijeljen na područja koji se mogu i ne moraju preklapati, a pripadnost svakom dijelu određuje se klasama. Na temelju izmjerenih podataka i pravila dobivenih iz skupa podataka korištenih za učenje (engl. *training data*) čvor određuje klase kojima pripada, te zatim posljedično kao presjek ili centroidu više područja i svoju konačnu lokaciju. U drugom slučaju traži se korelacija mjerenja samog čvora i prethodno izvedenih mjerenja na točno određenim lokacijama te se lokacija čvora određuje kao centroida točaka s najvećom korelacijom.

3.3 Srodna istraživanja

U ovom dijelu analizirati će se istraživanja koja su prema prethodno definiranim klasifikacijama i tipu mjerenja srodna izvornom algoritmu. Algoritam je već predstavljen u poglavlju 1.2, a detaljno će biti opisan u poglavlju 4. S obzirom na prethodno navedene klasifikacije to je algoritam koji spada u raspodijeljene algoritme i kao takav ne pretpostavlja postojanje čvorova koji poznaju svoju lokaciju odnosno funkcionira u neusidrenim mrežama. Iako se taj princip često primjenjuje za slučajeve kada se mjeri udaljenost

među čvorovima, ovo istraživanje je, prema dostupnoj literaturi, jedini primjer primjene ove metode na mreže kod kojih se mjeri azimut među čvorovima. To se vidi i u tablici 3.1 gdje su prikazana srodna istraživanja u kojima su korišteni algoritmi kod kojih se mjeri azimut. Iz tablice se može zaključiti kako je velika većina algoritama raspodijeljena i funkcionira u usidrenim mrežama te nemaju eksperimentalnu verifikaciju.

U skupini algoritama kod kojih je obavljena eksperimentalna verifikacija su algoritmi koji funkcioniraju samo u mrežama u kojima su svi čvorovi u dometu dovoljnog broja sidara, odnosno ne funkcioniraju u takozvanim *multihop* mrežama [28, 37, 38]. Ono što razlikuje ove algoritme je u prvom redu metoda mjerenja azimuta. Primjerice u [37] TripLoc algoritam se koristi takozvanim radiointerferencijskim mjerenjem (engl. *radio interferometric measurement - RIM*) čiji preduvjet su tri ortogonalne neusmjerene antene postavljene na sidrima. Dvije od njih emitiraju visokofrekvencijski signal s malom razlikom u frekvenciji kako bi njihovom interfencijom dobili niskofrekvencijski signal čiju se razliku u fazi može dovoljno točno mjeriti. Razlika u fazi niskofrekvencijskog signala se određuje između signala primljenog od strane treće antene i antene na čvoru kojem se određuje lokacija. Dobivena razlika je proporcionalna razlici udaljenosti između antena na sidru i antene na čvoru, a moguća rješenja za lokaciju čvora nalaze se na hiperboli određenoj tom razlikom pri čemu se hiperbola aproksimira asimptotama te se na taj način zapravo određuje azimut. U algoritmu ALRD [38] svako sidro je opremeljeno s dvije usmjerene antene čiji dijagrami zračenja su međusobno zakrenuti za 90° u ravnini u kojoj se određuje lokacija čvorova. Čvor kojem se određuje lokacija estimira azimut prema sidru s obzirom na razliku u jakosti signala koju primi sa svake od antena i poznatog dijagrama zračenja. Algoritam opisan u [28] koristi se sidrima opremeljenim rotirajućim laserima. Laseri su sinkronizirani u svojoj vrtnji te se na temelju razlike u vremenu detekcije laserske zrake na čvoru čija se lokacija određuje može izravno estimirati njegova lokacija.

Iz pricipa na kojem se zasniva ova skupina algoritama jasno je kako broj sidara, njihov raspored i domet mora biti takav da osigura da svaki čvor mreže bude u dometu barem dva sidra kao što je već spomenutu u poglavlju 3.2.1 i prikazano na slici 3.4. To je prilično veliki zahtjev s obzirom da se u ovisnosti o metodi mjerenja azimuta zahtjeva da sidra budu posebno opremljena. Dodatna otegotna okolnost su i veliki energetski zahtjevi sidara, pogotovo ukoliko je mreža nepravilne topologije i/ili ako ima veliki broj čvorova. Pozitivni aspekt ovog tipa određivanja lokacije je što čvorovi čija se lokacija određuje, u pravilu, lokaciju mogu obaviti brzo i energetski efikasno te oni sami, za razliku od sidara, ne moraju biti posebno opremljeni.

Sljedeća zanimljiva opservacija je da se uz algoritam iz ove disertacije, samo jedan algoritam koristi tehnikom lokaliziraj-i-spoji (engl. *Map & Stitch*) [39]. Međutim, taj

Tablica 3.1: Algoritmi za određivanje položaja mjerenjem azimuta.

Algoritam	**Tip mjerenja**	**Raspodijeljeno**	***Map&Stitch***	**Neusidreno**	***Multihop***	**Eksperimentalna verifikacija**
Izvorni algoritam	Azimut	DA	DA	DA	DA	DA
DAL [40]	Azimut	DA	DA[1]	DA	DA	NE
APS-AOA [29]	Azimut	DA	NE	NE	DA	NE
Probabilistic [41]	Azimut	DA	NE	NE	DA	NE
n/a [42]	Azimut	DA	NE	NE	DA	NE
n/a [43]	Azimut	DA	NE	NE	DA	NE
n/a [44]	Azimut	DA	NE	NE	DA	NE
AT-Angle [45]	Azimut	DA	NE	NE	DA	NE
ADNL-Angle [46]	Azimut	DA	NE	NE	DA	NE
TripLoc [37]	Azimut	DA	NE	NE	NE	DA
ALRD [38]	Azimut	DA	NE	NE	NE	DA
RotatingBeacon [19, 28]	Azimut	DA	NE	NE	NE	DA
AVPLE-WIV						
BCAVPLE-WIV [31]	Azimut	DA	NE	NE	NE	NE
RAST [47]	Azimut	NE	NE	DA	DA	NE
MaximumRigid [39]	Azimut	NE	DA	DA	DA	NE

[1]spajanje se obavlja tek u zadnjoj fazi između segmenata koji zauzimaju velike dijelove mreže i čija lokacija je prethodno određena iterativnom tirangulacijom

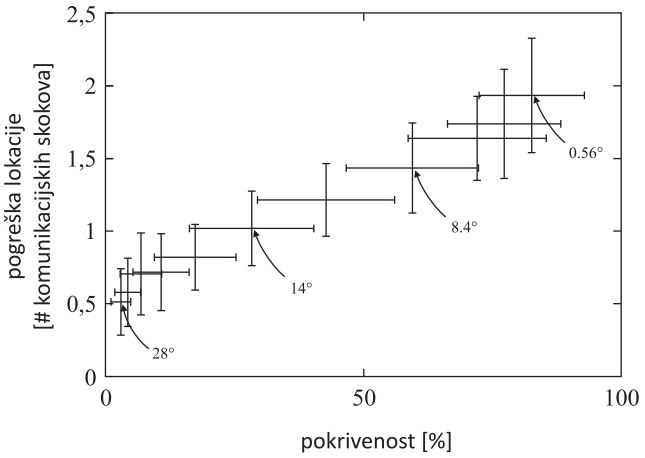
algoritam je fokusiran na pronalazak najvećih krutih komponenti u mrežama s velikim brojem čvorova i to, zbog zahtjevnosti zadatka, obavlja centralizirano. Također zaključak koji autori navode, a to je maksimalno iskorištenje mjernih podataka kako bi broj čvorova u istom koordinatnom sustavu bio najveći, ne uzima u obzir pogreške mjerenja i nepovoljne formacije čvorova, te u tim slučajevima postoji mogućnost nekvalitetnih estimacija, što se u radu ne spominje niti analizira.

Veliki broj algoritama funkcionira raspodijeljeno u *multihop* mrežama i zahtjeva postojanje sidara. Jedan od prvih i najpoznatijih algoritama APS-AOA [29] se temelji na principu vektora udaljenosti opisanog u poglavlju 3.2.4. Razlika je što se u ovom slučaju umjesto propagacije broja skokova ili udaljenosti, propagira azimut od pojedinog sidra do čvora. Algoritam ima slične nedostatke: točan je isključivo ukoliko je raspored čvorova homogen, a sidra se nalaze uz rubove mreže. Nakon objave pojavile su se inačice ovog algoritma s dodatnim optimizacijama, za posebne uvjete ili, u slučajevima u kojima postoje dodatne informacije poznate čvorovima, za njihovo bolje iskorištenje. Jedna od takvih inačica je opisana u [41], a zasniva se na propagaciji azimuta od sidara korištenjem velikog broja različitih međusobno nezavisnih puteva i održavanje zapisa o primljenim azimutima duž svakog puta. Nakon toga čvorovi za svaki put iz zapisa računaju funkciju gustoće vjerojatnosti (engl. *probability density function - pdf*) i naposlijetku računaju združeni pdf i iz njega estimiraju lokaciju. Nakon uspješno određene lokacije čvorovi sami postaju sidra. Prema rezultatima simulacija, algoritam se pokazao točniji od APS-AOA međutim uz povećanu složenost i memorijsko, računalno i komunikacijsko opterećenje, zahtjeva i točan model mjerenja azimuta uključujući i podatak o maksimalnom dometu osjetila.

Određeni broj raspodijeljenih algoritama se temelji na iterativnoj optimizaciji na lokalnoj razini [40, 42, 43]. Iako su matematičke metode koje se koriste u tim algoritmima različite, zajedničko je da se početne estimirane lokacije, koje mogu biti i slučajne, u svakom daljnjem koraku optimizacije ciljano pomiču u smjeru u kojem će se zadovoljiti lokalna mjerenja azimuta među čvorovima susjedima. S obzirom da su algoritmi raspodijeljeni, informacije na temelju kojih se donose odluke o smjeru pomaka su isključivo lokalne i sastoje se od estimiranog položaja čvorova susjeda u prethodnom koraku. U radovima su dodatno analizirani uvjeti u kojima algoritmi konvergiraju te brzina konvergencije pravom rješenju u vidu broja koraka ili broja paketa [42]. U [44] je obavljena i analiza osjetljivosti na pogreške u položaju sidara, i pogreške u mjerenju azimuta. Problem ovog tipa algoritama je što, iako su raspodijeljeni i izmjena podataka je samo na lokalnoj razini, komunikacija je intezivna i rezultira velikim količinom odaslanih podataka. Ono što je još nepovoljnije je veliki broj paketa, s obzirom da se po prirodi algoritma relativno mali broj podataka po paketu mora slati naizmjenično što može stvoriti

probleme u protokolima za pristup mediju.

Točnost estimiranih lokacija položaja ovisi o točnosti mjerenja, ali vrlo često je dominantniji izvor pogrešaka u estimaciji nepovoljan međusobni položaj čvorova. Primjer pogrešaka koje mogu nastati kod algoritama koji se služe mjerenjem udaljenosti već je prikazan na slici 2.6. Takve nepovoljne formacije, kao što je spomenuto u poglavlju 2.2, nisu globalno krute te onemogućavaju jedinstveno rješenje i posljedično uzrokuju velike pogreške preklapanja (engl. *flip error*). Iako kod algoritama koji se koriste mjerenjima azimuta ne postoji mogućnost pojave pogrešaka preklapanja, utjecaj nepovoljnih formacija može biti velik. Procjena tih pogrešaka za neusidrene mreže detaljno se analizira u idućem poglavlju. Bez obzira na tu činjenicu, tek nekoliko algoritama uzima ih u obzir i pokušava ograničiti njihovu propagaciju. Jedan od takvih algoritama je AT-Angle [45] kod kojeg se pogreška estimacije procjenjuje na temelju veličine područja u kojem bi se, prema algoritmu, čvor trebao nalaziti. Nadalje, kod već spominjanog algoritma APS-AOA, opisanog u [29], propagacija azimuta od sidara prema čvorovima mreže se obavlja samo ukoliko trokuti kojima se obavlja propagacija nemaju kuteve koji su manji od unaprijed definiranog praga. Na slici 3.9 se vidi kako veličina ovog praga ima utjecaj na točnost estimiranih lokacija i na udio čvorova čija je lokacija estimirana. Identičnom metodom praga koristi i algoritam DAL [40]



Slika 3.9: Pokrivenost i pogreška lokacije APS-AOA [29] algoritma za različite veličine praga za najmanji iznos kuta trokuta uz pomoć kojih se obavlja propagacija azimuta.

DAL je, od prethodno navedenih algoritama koji se koriste mjerenjima azimuta, jedini koji je raspodijeljen i funkcionira u neusidrenim mrežama. Algoritam za zasniva na iterativnoj triangulaciji i dijeli se u nekoliko faza. U prvoj fazi se definiraju primarni grozdovi gdje svaki čvor odredi skup susjeda koji su vezani u tzv. lance trokuta i u kojima čvorovi imaju jedinstvenu lokaciju. Nakon toga odvija se iterativno dodavanje čvorova iz primarnog podgrozda s manjim jedinstvenim identifikacijskim brojem (ID-em) u onaj s

većim. Kada završi ova faza, u mreži ostaju samo primarni podrgrozdovi koje algoritam pokušava spojiti u trećoj fazi prema kriteriju broja čvorova u podgrozdu. Problem s ovakvim pristupom je što se u drugoj ključnoj fazi koristi samo dio dostupnih podataka s obzirom da se lokacija čvora određuje iz samo jedne kombinacije relativnih azimuta odnosno jednog trokuta. Autori u tom slučaju koriste već spomenutu heuristiku prema kojoj se ne koriste trokuti koji imaju jedan kut manji od praga koji u njihovom slučaju iznosi $4°$. Drugi problem je što se za određivanje duljine bridova koristi heuristika prema kojoj čvorovi koji pripadaju različitim podgrozdovima moraju biti udaljeniji od onih koji pripadaju istim što ne mora biti zadovoljeno u mrežama s nekonveksim topologijama ili u mrežama koje se nalaze u prostorima s puno prepreka. Dodatno, autori u realno očekivanom slučaju u kojem postoji šum prilikom mjerenja azimuta uvode dodatnu fazu koja se koristi već spomenutom iterativnom optimizacijom na lokalnoj razini s, također već spomenutim, nedostatkom u vidu prevelikog broja poruka.

Algoritmi koji se koriste mjerenjima udaljenosti su puno prisutniji u literaturi i to je područje do sada općenito primilo više pažnje. Zbog većeg broja istraživanja u tom području se nailazi i na algoritme koji koriste *lokaliziraj-i-spoji* tehniku te pokušavaju kontrolirati propagaciju pogrešaka zbog nepovoljnih formacija. Pregled takvih algoritama koji su po tim metodama srodni algoritmu iz ove disertacije dan je u tablici 3.2.

Pregled započinjemo algoritmima koji se zasnivaju na metodi MDS opisanoj u dijelu 3.2.2. Algoritmi lokalizacije temeljeni na MDS-u su u osnovi centralizirani algoritmi no razvijene su i njihove raspodijeljene inačice koje se u većini slučajeva koriste već spomenutom tehnikom podijele i spajanja. Jedan od prvih predstavnika ove metode je algoritam MDS-MAP u inačicama (P,R) [48] i (D) [2]. MDS-MAP(P,R) u prvoj fazi određuje članove lokalnih formacija za svaki čvor prema kriteriju maksimalne udaljenosti po broju komunikacijskih skokova R_{lm} (u radu autori koriste $R_{lm} = 2$). Zatim se u drugoj fazi, koristeći potpunu matricu izmjerenih i estimiranih udaljenosti, određuju lokacije svih čvorova, te se obavlja lokalna optimizacija metodom najmanjih kvadrata. U trećoj fazi se odvija spajanje pri čemu je redoslijed određen isključivo prema kriteriju najvećeg broja zajedničkih čvorova. Nakon spajanja, u četvrtoj fazi se odvija opcionalna globalna optimizacija koju autori označavaju kao računalno najskuplji dio algoritma i koji jedini nije raspodijeljen. Na kraju, ukoliko postoje sidra, cijeli koordinatni sustav se transformira kako bi se uskladio s poznatim pozicijama sidara. Algoritam se dobro nosi s nepravilnim topologijama, međutim u radu se spominje nedovoljno dobro ponašanje kada je povezanost mreže mala što se pripisuje greškama prilikom određivanja lokacija u lokalnim mapama u drugoj fazi, te propagaciju istih prilikom spajanja u trećoj fazi algoritma. Isti autori u (D) inačici algoritma umjesto četvrte faze odnosno globalne centralizirane optimizacije uvode distribuiranu optimizaciju koristeći, već spomenutu,

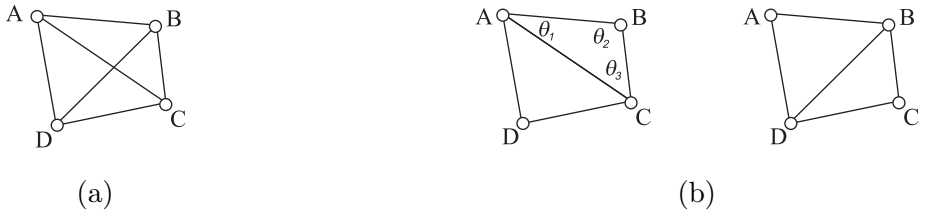
Tablica 3.2: Algoritmi za određivanje položaja mjerenjem udaljenosti.

Algoritam	**Tip mjerenja**	**Raspodijeljeno**	***Map&Stitch***	**Neusidreno**	***Multihop***	**Eksperimentalna verifikacija**
MDS-MAP(P,R) [48]	Udaljenost	DA[1]	DA	DA	DA	NE
MDS-MAP(D) [2]	Udaljenost	DA	DA	DA	DA	NE
RobustQuad [1]	Udaljenost	DA	DA[2]	DA	DA	DA[3]
ASAP [3]	Udaljenost	DA	DA	DA	DA	NE
MCF, MaxMin [4]	Udaljenost	DA	DA	DA	DA	NE
FR-MDS [7]	Udaljenost	DA	DA	DA	DA	NE
ETOC [5]	Udaljenost	DA	DA	NE	DA	DA
SPG-L/G, 2Pass [49]	Udaljenost	DA	DA	NE	DA	NE

[1]Faza u kojoj se obavlja dodatno usavršavanje položaja se ne obavlja raspodijeljeno.
[2]Nije opisano na koji način i kojim redoslijedom se obavlja spajanje
[3]Eksperimentalno je verificiran samo primjer praćenja mobilnog čvora unutar područja pokrivenog mrežom s čvorovim s poznatom lokacijom.

iterativnu optimizaciju na lokalnoj razini.

Tip pogreške koji najviše utječe na točnost prilikom spajanja formacija i koji je uzrok spomenutom slabim rezultatima kod mreža s malom povezanosti je sličan pogrešci koja je već opisana i prikazana na slici 2.6b. Naime, ukoliko zajednički čvorovi između segmenata koji se spajaju tvore formaciju koja nije ispravno orijentirana, jedan od segmenata se prilikom tog spajanja reflektira na pogrešnu stranu. Ta pogreška naziva se pogreška preklapanja (engl. *flip error*) i prikazana je na slici 3.11.



(a) (b)

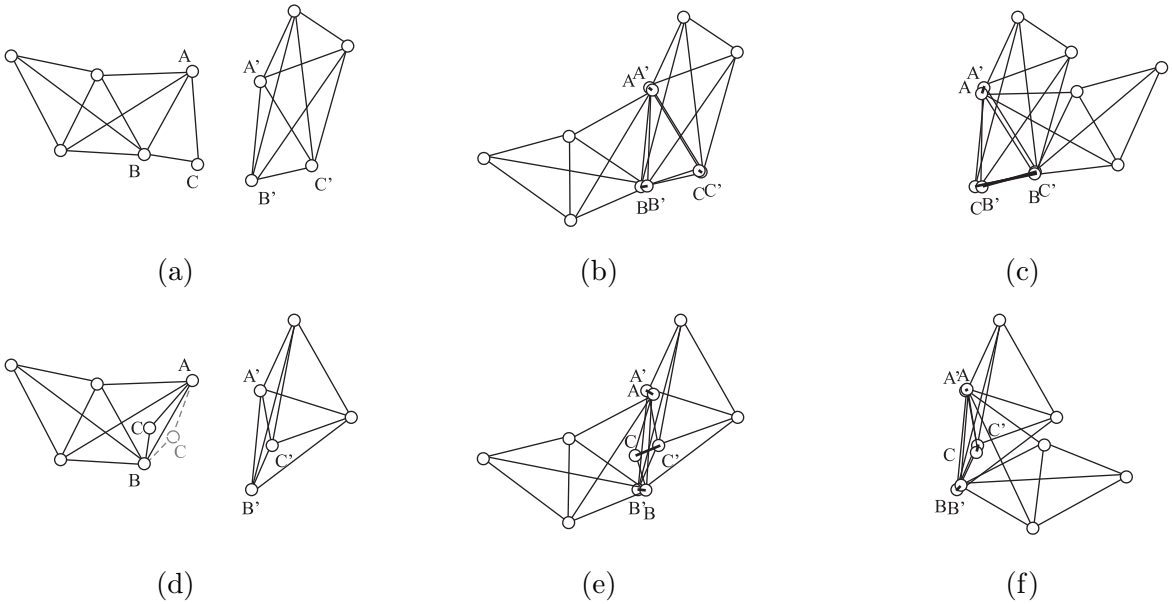
Slika 3.10: Robusni četverokut (a) [1] sastoji se od četiri trokuta (b) čiji najmanji kutevi θ i najkraće stranice b zadovoljavaju jednadžbu (3.8).

Pregled algoritama koji pokušavaju riješiti ovaj problem započinjemo s algoritmom RobustQuad [1] koji se temelji na procjeni i minimizaciji mogućnosti pojave preklapanja prilikom samog inicijalnog određivanja lokacija. Princip je vrlo jednostavan: jedina dopuštena formacija čvorova u početnoj fazi u kojoj se određuju lokacije inicijalnih segmenata je tzv. robusni četverokut (engl. *rubust quadrilateral*) prikazan na slici 3.10a. To je formacija od četiri potpuno povezana čvora odnosno, gledajući iz druge perspektive, koja se sastoji od četiri trokuta prikazana na slici 3.10b. Kako bi se ova formacija mogla nazvati robusnom na pogreške preklapanja autori izvode uvjet koji moraju zadovoljavati sva četiri trokuta:

$$b \sin^2(\theta) > d_{\min} \tag{3.8}$$

pri čemu je b duljina najkraće stranice trokuta, a θ najmanji kut. $d_{\min}$ je parametar algoritma kojeg se može izabrati u ovisnosti o standardnoj devijaciji mjerenja udaljenosti te o željenoj vjerojatnosti pojave pogreške preklapanja. Algoritam je u osnovi jednostavan i funkcionira upravo onako kako su autori i predvidjeli te vrlo uspješno izbjegava pogrešku preklapanja. Glavni nedostatak mu je što je za mreže s manjom povezanosti i većom standardnom devijacijom mjerenja postotak čvorova kojima se odredi lokacija izrazito nizak što se može vidjeti na slici 2.8.

Sljedeći primjer algoritma koji pokušava riješiti problem pogrešnog preklapanja je opisan u [7]. Algoritam detektira pogreške preklapanja na lokalnoj i globalnoj razini. Na lokalnoj razini prilikom spajanja dvije formacije ona prava se određuje prema kriteriju sume udaljenosti opisanom na slici 3.11. Ukoliko sumu udaljenosti u slučaju bez refleksije označimo s E_1, a sa refleksijom s E_2 tada je, kako bi spajanje bilo dovedeno u



Slika 3.11: Pogreška preklapanja prilikom spajanja dviju formacija. Za formacije na slikama (a) i (d) moguće je obaviti spajanje bez refleksije lijeve formacije kao na slikama (b) i (e) ili s refleksijom kao na slikama (c) i (f). U prvom primjeru prikazanom na slikama (a), (b) i (c) jednostavno je donijeti pravu odluku i izabrati inačicu bez refleksije kao pravu s obzirom slaganje čvorova ABC i $A'B'C'$ nakon spajanja koristeći kriterij sume duljina između lokacija AA', BB' i CC'. U drugom slučaju su lokacije čvorova ABC takve da relativno mala pogreška u lokaciji čvora C (prava lokacija je označena iscrtkanom kružnicom) može dovesti do situacije u kojoj se po istom kriteriju odabire pogrešna, odnosno reflektirana, verzija formacije što u konačnici rezultira velikim pogreškama u lokacijama svih ostalih čvorova u tom segmentu i daljnjoj propagaciji te pogreške.

razmatranje, potrebno zadovoljiti sljedeća dva kriterija:

$$\begin{aligned} \min\{E_1,E_2\} &< \text{THRESH\_SUM} \\ \frac{\max\{E_1,E_2\}}{\min\{E_1,E_2\}} &> \text{THRESH\_SUM} \end{aligned} \tag{3.9}$$

pri čemu su THRESH_SUM i THRESH_RATIO parametri algoritma.

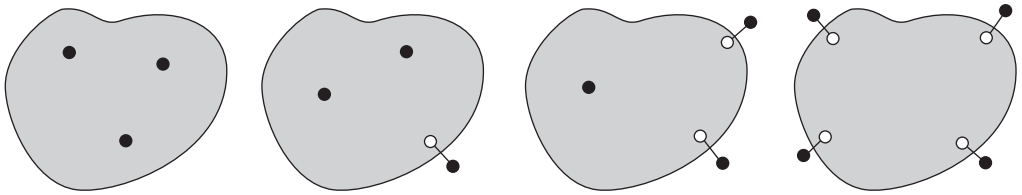
Nakon što se na ovaj način detektiraju i filtriraju sva lokalna spajanja koja imaju veću vjerojatnost pogreške preklapanja, algoritam stvara poseban graf preklapanja (engl. *flip graph*) u kojem se vrhovi lokalne formacije koje je potrebno spojiti, a bridovi su operacije spajanja koja među tim formacijama treba obaviti. Bridovima tog grafa se dodjeljuje informacija ($I = 0$ ili $I = 1$) koja označava je li među segmentima predstavljenim vrhovima incidentnim tom bridu potrebna refleksija jednog od njih ili nije. Nakon toga se otkrivaju konflikti u grafu preklapanja odnosno situacije kada za istu formaciju nije jednoznačno određeno reflektira li se ili ne neovisno o redoslijedu spajanja. Posebnim raspodijeljenim algoritmom detekcije konflikata se definira najveći segment koji je konzistentan s obzirom na vrijednosti dodijeljene bridovima u grafu preklapanja. Slična metoda koristi

se i u algoritmu ASAP [3] gdje se stvara matrica refleksija $\mathcal{Z}$ s vrijednostima $z_{ij} = -1$ u slučaju kada je potrebna refleksija jedne od formacija i ili j, $z_{ij} = 0$ kada formacije nisu povezane ili $z_{ij} = 1$ kada nije potrebna refleksija. Za rješavanje konflikata ASAP se koristi spektralnom dekompozicijom matrice $\mathcal{Z}$ koja se također može obaviti na raspodijeljeni način te estimacijom potrebnih refleksija $\hat{z}_i$ iz svojstvenog vektora koji odgovara najvećoj svojstvenoj vrijednosti. Naposlijetku, i u jednom i u drugom slučaju obavlja se spajanje svih formacija u tom segmentu koristeći informacije o potrebnim refleksijama I odnosno $\hat{z}_i$.

Iako algoritam garantira nepostojanje konflikata bez obzira na poredak spajanja, o poretku spajanja može ovisiti kvaliteta rezultata što isti autori analiziraju u [49]. U tom radu analizira se i formalizira nekoliko heuristika kroz definiciju politike spajanja (engl. *stitching policy*) i funkcije kvalitete spajanja (engl. *potential function*). Definiraju se dvije politike spajanja. Prva je jednoprolazna pohlepna (engl. *single-pass greedy – SPG*) u kojoj se prije svakog spajanja na globalnoj ili lokalnoj razini (postoje dvije inačice SPG-G i SPG-L) određuje formacija koja ima najveći iznos funkcije kvalitete spajanja odnosno koja se sljedeća spaja. Druga je dvoprolazna (engl. *two-pass*) u kojoj se u prvom prolazu određuje poredak u obliku usmjerenog acikličkog grafa (engl. *directed acyclic graph – DAG*), a u drugom obavlja spajanje. Funkcija kvalitete spajanja svakoj formaciji, koja se potencijalno može spojiti, pridružuje iznos koji ovisi o tri faktora: (1) kvaliteti prethodnih spajanja – što čini funkciju rekurzivnom, (2) korelaciji između dvije formacije koje se spajaju i (3) procjeni kvalitete formacije koja se treba spojiti. Autori u radu analiziraju nekoliko funkcija koje koriste prva dva faktora, međutim ne analiziraju nijednu koja koristi treći, odnosno procjenu kvalitete lokacija unutar segmenta koji se treba spojiti. Naime, navodi se kako su simulacije pokazale da je utjecaj ovog faktora zanemariv, no za procjenu kvalitete su se koristili metodom koja uzima u obzir samo iznose mjerenja, a ne i njihov broj te dobivenu formaciju. Ispravan način procjene kvalitete određivanja lokacija unutar formacija je jedan od doprinosa ovog rada koji se detaljno analizira u narednim poglavljima za slučaj mjerenja azimuta. Simulacije pokazuju kako najkvalitetnije rezultate daje SPG-G politika spajanja koja se, s obzirom na svoju globalnu prirodu, ne može učiniti raspodijeljenom, pa autori predlažu korištenje SPG-L čiji rezultati iako lošiji još uvijek predstavljaju dobar kompromis.

Jedan od recentnih algoritama koji se koristi tehnikom lokaliziraj-i-spoji za mreže kod kojih se mjeri udaljenost među čvorovima je ETOC [5]. U prve dvije faze algoritam ne nudi osobite novosti i napredak u odnosu na postojeća rješenja: za inicijalne formacije se koristi iterativna trilateracija, a spajanje u veće komponente se odvija koristeći čvorove zajedničke za obje formacije. Pomak je napravljen detaljnom analizom svih situacija (slika 3.12) u kojima je nakon određivanja lokacija u relativnom koordinatnom

sustavu moguće integrirati podatak o sidrima. Koristeći tu analizu, autori predlažu metode kojima se nakon identifikacije tipa situacije mogu izračunati lokacije u apsolutnom koordinatnom sustavu što posebice dolazi do izražaju u mrežama s malom povezanošću. Naposlijetku, ETOC je trenutno jedini algoritam iz ove skupine koji uz simulacije ima i eksperimentalnu verifikaciju.



Slika 3.12: Lokacije sidara u odnosu na krutu komponentu u kojima je moguće odrediti apsolutne lokacije sidara u komponenti [5]: (a) tri ili više sidara unutar komponente, (b) dva sidra, (c) jedno sidro i (d) bez sidara unutar komponente.

Poglavlje 4

Metoda procjene točnosti estimiranih lokacija u neusidrenim mrežama

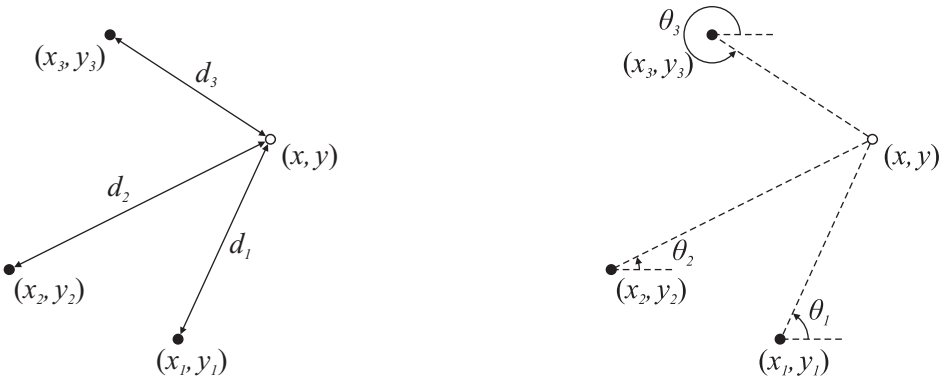
U prethodnim poglavljima analizirani su algoritmi za određivanje položaja koji su korišteni i ispitani u recentnim istraživanjima odnosno u pripadajućoj literaturi. Većina strategija koristi različite heuristike kako bi se kontrolirala i ograničila propagacija pogrešaka. Kod raspodijeljenih algoritama, u mrežama s više skokova, ta propagacija može imati veliki utjecaj na točnost estimiranih lokacija. Autori u [49] predlažu korištenje procjene točnosti estimiranih lokacija kao jedan od podataka koji se može koristiti kao ulazni parametar za heuristike za kontrolu propagacije pogrešaka. Međutim ta procjena se u navedenom radu izvodi jednostavnom usporedbom izmjerenih udaljenosti i udaljenosti dobivenih iz estimiranih lokacija. Procjena obavljena na taj način ne uzima u obzir relativan raspored čvorova koji često predstavlja dominantan uzrok nedovoljno kvalitetne estimacije, pa u nastavku rada autori koriste i analiziraju ostale predložene parametre.

Osnovna hipoteza ovog dijela istraživanja je da se u algoritmima za određivanje lokacija može koristiti procjena točnosti estimacije lokacija koja se dobiva iz samih estimiranih koordinata. U dostupnoj literaturi iz područja nije provedena takva analiza, niti je opisana metoda kojom bi se to učinilo. Stoga se u ovom poglavlju najprije analiziraju postojeće metode za procjenu kvalitete estimacije lokacija koje uzimaju u obzir relativan položaj čvorova. Detaljno se izvode metode i izrazi za procjenu te navode pretpostavke za njihovo korištenje. Nakon toga se analizira mogućnost upotrebe u algoritmima za raspodijeljeno određivanje položaja u neusidrenim mrežama, te se detaljno opisuje metoda za tu namjenu. Naposljetku, predložena se metoda implementira u simulatoru te koristi za estimaciju kvalitete određivanja lokacija u inicijalnim podgrozdovima. Dobiveni rezultati se analiziraju s naglaskom na traženu korelaciju između procijenjene kvalitete i

točnog, čvoru nepoznatog, iznosa pogreške estimacije lokacija.

4.1 Estimacija lokacije metodom najmanjih kvadrata u usidrenim mrežama

Kako bi se dobio potpuni uvid u procjenu kvalitete estimacije, u narednom dijelu opisana je generička metoda estimacije lokacije u usidrenim mrežama [9]. Pretpostavka je kako postoji M čvorova sidara i jedan čvor čiju lokaciju estimiramo na osnovu M mjerenja μ_i, $i = 1,2,\ldots,M$ geometrijskog odnosa sidara i samog čvora (slika 4.1). Vektor $\boldsymbol{p} = [p_1,p_2,\ldots,p_n]$ duljine n predstavlja vektor čije vrijednosti se estimiraju. Obično su to koordinate, a n je u tom slučaju dimenzija prostora. Ukoliko algoritam koristi dodatne veličine koje je potrebno estimirati, npr. odmak sata između satelita i prijamnika kod sustava za globalno pozicioniranje (engl. *Global Positioning System – GPS*), i one, također, mogu postati dio vektora $\boldsymbol{p}$.



(a) $\mu = d$ (b) $\mu = \theta$

Slika 4.1: Estimacija lokacije (x,y) koristeći $M = 3$ mjerenja μ između sidara, označenih sa $\bullet$, i čvora $\circ$, pri čemu je na slici (a) prikazano mjerenje udaljenosti, a na slici (b) azimuta.

Mjerenja μ_i podložna su pogreškama. Uz pretpostavku umjeravanja odnosno anuliranja sustavne pogreške, svako se mjerenje može smatrati sumom poznate funkcije $f_i(\boldsymbol{p})$ koja povezuje lokaciju i geometrijski odnos koji se mjeri, te slučajne varijable e koja predstavlja mjernu pogrešku: $\mu_i = f_i(\boldsymbol{p}) + e, i = 1,2,\ldots,M$. Nadalje, M takvih jednadžbi mogu se napisati u vektorskom obliku kao:

$$\boldsymbol{\mu} = \boldsymbol{f}(\boldsymbol{p}) + \boldsymbol{e}. \tag{4.1}$$

Ako se vektor pogreške mjerenja $\boldsymbol{e}$ modelira normalnom razdiobom sa srednjom vrijednosti 0 i uz matricu kovarijanci $\boldsymbol{\Sigma}$, tada funkcija vjerodostojnosti (engl. *likelihood*

function) mjerenja poprima sljedeći oblik:

$$p(\boldsymbol{\mu}|\boldsymbol{p}) = \frac{1}{(2\pi)^{M/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-(1/2)[\boldsymbol{\mu}-\boldsymbol{f}(\boldsymbol{p})]^T \boldsymbol{\Sigma}^{-1}[\boldsymbol{\mu}-\boldsymbol{f}(\boldsymbol{p})]}. \tag{4.2}$$

Jednadžba (4.2) predstavlja funkciju gustoće uvjetne vjerojatnosti mjerenja $\boldsymbol{\mu}$ za dani vektor lokacije $\boldsymbol{p}$. Estimator lokacije odnosno vektora $\boldsymbol{p}$ po maksimalnoj vjerodostojnosti (engl. *maximum likelihood estimator – MLE*) je ona vrijednost $\boldsymbol{p}$ u kojoj funkcija (4.2) postiže svoj maksimum. S obzirom na oblik funkcije, maksimum se postiže minimiziranjem kvadratne forme:

$$Q(\boldsymbol{p}) = (\boldsymbol{\mu} - \boldsymbol{f}(\boldsymbol{p}))^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{f}(\boldsymbol{p})). \tag{4.3}$$

Kako je funkcija koja povezuje lokaciju i mjerenje $\boldsymbol{f}(\boldsymbol{p})$ u pravilu nelinearna, u cilju pojednostavljenja estimatora ona se linearizira razvojem u Taylorov red i zadržavanjem prva dva člana:

$$\boldsymbol{f}(\boldsymbol{p}) \simeq \boldsymbol{f}(\boldsymbol{p}_0) + \boldsymbol{G}(\boldsymbol{p} - \boldsymbol{p}_0). \tag{4.4}$$

pri čemu je $\boldsymbol{G}$ Jakobijeva matrica funkcije mjerenja u točki $\boldsymbol{p}_0$ veličine $M \times n$:

$$\boldsymbol{G} = \begin{bmatrix} \frac{\partial f_1}{\partial p_1}\Big|_{\boldsymbol{p}=\boldsymbol{p}_0} & \cdots & \frac{\partial f_1}{\partial p_n}\Big|_{\boldsymbol{p}=\boldsymbol{p}_0} \\ \vdots & & \vdots \\ \frac{\partial f_M}{\partial p_1}\Big|_{\boldsymbol{p}=\boldsymbol{p}_0} & \cdots & \frac{\partial f_M}{\partial p_n}\Big|_{\boldsymbol{p}=\boldsymbol{p}_0} \end{bmatrix}. \tag{4.5}$$

Izbor točke $\boldsymbol{p}_0$ može proizaći iz prethodnog koraka iterativne procedure, a u narednom dijelu se pretpostavlja da je taj izbor dovoljno blizu pravoj vrijednosti $\boldsymbol{p}$, te kako je pogreška zbog linearizacije mala.

Linearizacijom kvadratna forma poprima pojednostavljeni oblik:

$$Q(\boldsymbol{p}) = (\boldsymbol{\mu}_1 - \boldsymbol{G}\boldsymbol{p})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{G}\boldsymbol{p}) \tag{4.6}$$

pri čemu je $\boldsymbol{\mu}_1 = \boldsymbol{\mu} - \boldsymbol{f}(\boldsymbol{p}_0) + \boldsymbol{G}\boldsymbol{p}_0$. Kako bi smo pronašli ML estimaciju, potrebno je derivirati oblik kvadratne forme s lineariziranom funkcijom mjerenja po vektoru $\boldsymbol{p}$ i izjednačiti ga s $\mathbf{0}$:

$$\nabla_{\boldsymbol{p}} Q(\boldsymbol{p}) = \begin{bmatrix} \frac{\partial Q}{\partial p_1} & \frac{\partial Q}{\partial p_2} & \cdots & \frac{\partial Q}{\partial p_n} \end{bmatrix}^T = \mathbf{0}. \tag{4.7}$$

Prije deriviranja potrebno je srediti kvadratnu formu:

$$\begin{aligned} Q(\boldsymbol{p}) &= (\boldsymbol{\mu}_1 - \boldsymbol{G}\boldsymbol{p})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{G}\boldsymbol{p}) \\ &= \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G}\boldsymbol{p} - (\boldsymbol{G}\boldsymbol{p})^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + (\boldsymbol{G}\boldsymbol{p})^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G}\boldsymbol{p} \\ &= \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - (\boldsymbol{G}^T (\boldsymbol{\Sigma}^{-1})^T \boldsymbol{\mu}_1)^T \boldsymbol{p} - \boldsymbol{p}^T \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \boldsymbol{p}^T \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G}\boldsymbol{p} \\ &= \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - (\boldsymbol{p}^T \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1)^T - \boldsymbol{p}^T \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \boldsymbol{p}^T \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G}\boldsymbol{p} \\ &= \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - 2\boldsymbol{p}^T \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \boldsymbol{p}^T \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G}\boldsymbol{p} \end{aligned} \tag{4.8}$$

Pri sređivanju su se koristile činjenice da je $\boldsymbol{\Sigma}$ simetrična matrica, kao i da je $\boldsymbol{p}^T \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1$ skalar, pa je u oba slučaja transponirana vrijednost jednaka originalu. Sada je jednostavno pronaći prvu derivaciju:

$$\nabla_{\boldsymbol{p}} Q(\boldsymbol{p})\big|_{\boldsymbol{p}=\hat{\boldsymbol{p}}} = 0 - 2\boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + 2\boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G}\hat{\boldsymbol{p}} = \boldsymbol{0} \tag{4.9}$$

te estimaciju $\hat{\boldsymbol{p}}$:

$$\begin{aligned} \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G}\hat{\boldsymbol{p}} &= \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 \\ \hat{\boldsymbol{p}} &= (\boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G})^{-1} \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 \\ &= \boldsymbol{p}_0 + (\boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G})^{-1} \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \left[\boldsymbol{\mu} - \boldsymbol{f}(\boldsymbol{p}_0)\right] \end{aligned} \tag{4.10}$$

Jednadžba (4.10) predstavlja varijantu Newton-Raphson iterativne metode [50]. Prikladnije se može formulirati i kao odnos pomaka $\Delta\boldsymbol{p}$ prema estimaciji u sljedećem koraku i razlike stvarnih i očekivanih mjerenja $\Delta\boldsymbol{\mu}$ za rješenje u trenutnom koraku:

$$\Delta\hat{\boldsymbol{p}} = (\boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G})^{-1} \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \Delta\boldsymbol{\mu} \tag{4.11}$$

Izraz $(\boldsymbol{G}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{G})^{-1} \boldsymbol{G}^T \boldsymbol{\Sigma}^{-1}$ predstavlja generalizirani inverz Jakobijeve matrice funkcije mjerenja za rješenje $\hat{\boldsymbol{p}}$ u trenutnom koraku.

4.2 Procjena točnosti estimiranih lokacija u usidrenim mrežama

Kako bi se procijenila kvaliteta estimacije položaja, potrebno je definirati izvore pogrešaka. U dosadašnjem tekstu spomenuta je mjerna pogreška kod određivanja geometrijskog odnosa među čvorovima (npr. udaljenost, azimut). Metoda za određivanje položaja je neovisna od mjerne metode, pa se mjerna pogreška smatra zadanim, nepromjenjivim ulaznim parametrom algoritma. Stoga je i u slučaju kada se procjenjuje kvaliteta esti-

miranih položaja mjerna pogreška tretirana na isti način. Osim što se smatra zadanom, ona se u velikoj većini slučajeva uzima i kao homogena, odnosno jednaka za sve entitete koji sudjeluju u određivanju položaja. Slučajevi u kojima se kvaliteta mjerenja može estimirati zavrjeđuju posebnu analizu, a u literaturi se obrađuju izdvojeno, kao specijalni slučajevi [51] te ih se u ovom radu neće detaljnije analizirati.

Sljedeći faktori koji utječu na kvalitetu estimacije položaja su domet mjerne opreme u kombinaciji s rasporedom čvorova odnosno njihova gustoća. Ovaj faktor se naziva povezanost (engl. *connectivity*) ili stupanj mreže, a definiran je kao prosječan broj čvorova s kojim pojedini čvor može izmjeriti geometrijski odnos. Međutim, utjecaj navedenih faktora nemoguće je u cijelosti obuhvatiti jednim brojem s obzirom da, osim same povezanosti, na kvalitetu estimiranih lokacija utječe i relativni raspored čvorova koji određuju položaj odnosno kojima se određuje položaj.

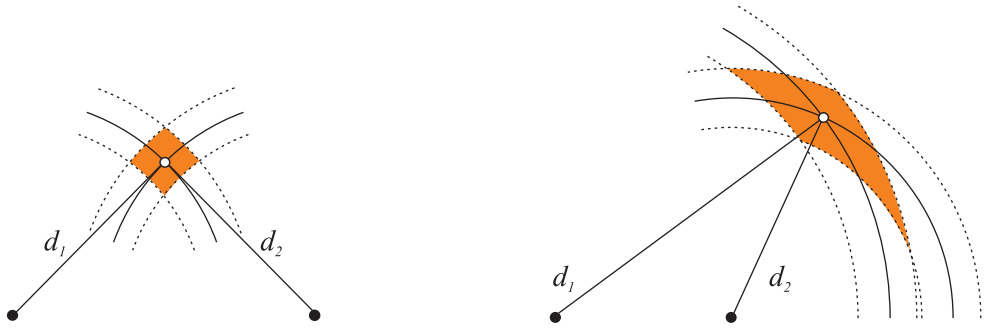
4.2.1 Geometrijsko slabljenje preciznosti za sustav globalnog pozicioniranja (GPS)

Jedna od osnovnih metoda za procjenu kvalitete određivanja položaja je korištenje faktora geometrijskog slabljenja preciznosti (engl. *geometric dilution of precision – GDOP*) koji, prema definiciji, predstavlja omjer standardnih devijacija estimacije položaja i estimacije geometrijskog odnosa:

$$\mathrm{GDOP} = \frac{\Delta\text{lokacija}}{\Delta\text{mjerenje}} = \frac{\sigma_p}{\sigma_\mu}. \tag{4.12}$$

Dijeljenjem standardnom devijacijom mjerenja σ_μ standardna devijacija estimacije položaja se normalizira u odnosu na mjernu pogrešku, te se anulira njen utjecaj na iznos faktora. Dobiveni izraz predstavlja upravo ovisnost kvalitete estimacije lokacije u odnosu na međusoban položaj entiteta koji određuju lokaciju. Time se jednim brojem kvantificira prethodno spomenuti utjecaj kombinacije dometa mjerne opreme i rasporeda čvorova na kvalitetu estimacije lokacije. Za slučaj estimacije lokacija iz izmjerenih udaljenosti između sidara i čvora kojem se određuje lokacija ovaj faktor se može intuitivno predstaviti kao na slici 4.2.

Jedan od najpoznatijih primjera korištenja ove metode procjene kvalitete estimacije je u sustavu globalnog pozicioniranja – GPS. Sustav se sastoji trenutno od 32 satelita s poznatim lokacijama (sidra) i preciznim i sinkroniziranim satovima od kojih barem četiri moraju biti vidljiva odnosno u dometu prijamnika kojem se određuje lokacija. Prijamnik mjeri lokalno vrijeme dolaska signala sa svakog od satelita t_{li} te uz primljeni podatak o točnim koordinatama satelita (x_i, y_i, z_i) i točnom vremenu slanja signala t_i računa udaljenost. Lokalni sat prijamnika i satelita nisu sinkronizirani, pa postoji ne-



(a) (b)

Slika 4.2: Geometrijsko slabljenje preciznosti u slučaju kada čvor kojem se određuje lokacija, označen s ∘, mjeri udaljenost od čvorova sidara, označenih s •, s jednakom mjernom pogreškom. U slučaju (a) lokacija čvora je takva da je faktor geometrijskog slabljenja preciznosti minimalan što se očituje malom površinom područja u kojem bi se čvor trebao nalaziti s obzirom na mjernu pogrešku. Na slici (b) GDOP poprima primjetno veće vrijednosti s obzirom na nepovoljnu lokaciju čvora u odnosu na sidra.

poznati vremenski odmak b koji je također potrebno estimirati. Kako su satovi satelita precizno sinkronizirani, taj odmak je neovisan o satelitu, odnosno jednak za sve satelite. Jednadžba za svaki od $M \geq 4$ satelita glasi:

$$(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 = [(t_{li} + b - t_i)c]^2, \quad i = 1,2,\ldots,M \tag{4.13}$$

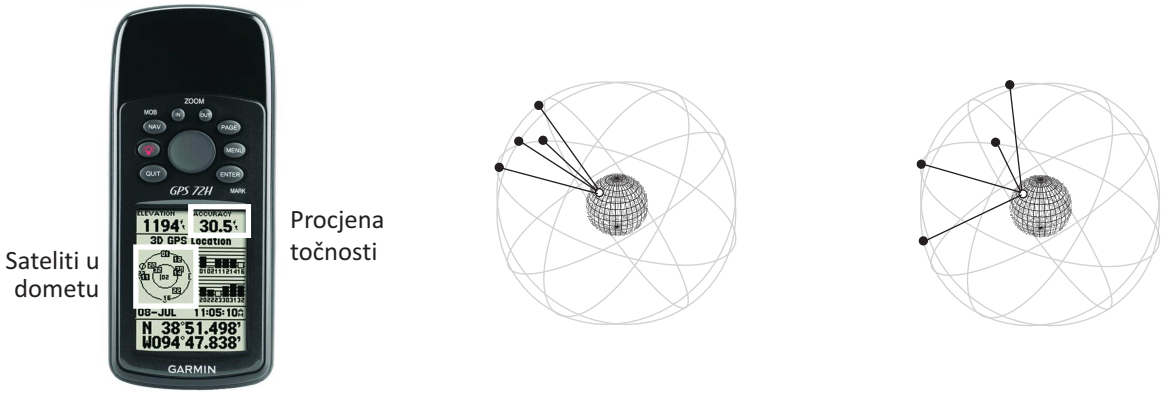
gdje je c poznata brzina propagacije signala. Ako se jednadžba izrazi u obliku udaljenosti od satelita d_i bez uračunatog vremenskog pomaka (tzv. pseudoudaljenost), dobiva se jednadžba mjerenja:

$$d_i = (t_{li} - t_i)c = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - bc \tag{4.14}$$

Ovaj sustav jednadžbi rješava se metodom najmanjih kvadrata pri čemu se estimiraju četiri nepoznanice, lokacija: x, y, z i vremenski odmak lokalnog sata b u formi vektora $\hat{\boldsymbol{p}} = [x, y, z, b]$. Optimizacijska jednadžba glasi:

$$\hat{\boldsymbol{p}} = \underset{\hat{\boldsymbol{p}}}{\operatorname{argmin}} \sum_i \left(\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - (t_{li} + b - t_i)c \right)^2 \tag{4.15}$$

Na slici 4.3a prikazan je tipičan GPS uređaj s označenom estimacijom pogreške. Estimacija pogreške uzima u obzir dva parametra, a to su pogreška mjerenja udaljenosti (engl. *User Equivalent Range Error – UERE*) i GDOP. Oba parametra su vremenski promjenjiva. UERE ovisi o slučajnim varijacijama u propagaciji signala sa satelita, i mjernim metodama u samom prijamniku, a promatran u duljem periodu slijedi normalnu



(a) (b) (c)

Slika 4.3: (a) GPS prijamnik na kojem je označen rezultat procjene točnosti estimacije lokacije koja kao ulazne parametre uzima pogrešku mjerenja udaljenosti – UERE i geometrijsko slabljenje preciznosti – GDOP. (b) Primjer formacije satelita u kojoj je GDOP velik i (c) u kojoj je malen.

razdiobu i za svaki satelit u prosjeku je jednak. Iznos GDOP-a varira jer ovisi o lokaciji satelita u dometu i samog prijamnika odnosno njihovom međusobnom odnosu koji se neprestano mijenja. GDOP je različit u horizontalnoj (HDOP) i vertikalnoj (VDOP) ravnini gdje je u prosjeku 1.5 puta veći [52]. Za razliku od pogreške mjerenja, promatrajući ga u duljem periodu, ne slijedi normalnu razdiobu, te može izrazito varirati u iznosu. Prijamnik kontinuirano računa GDOP satelitâ i na osnovu njega može u danom trenutku izabrati optimalnu kombinaciju od 4 satelita koje će koristiti. Kod izračuna GDOP-a može se uzeti u obzir i estimacija UERE-a svakog satelita posebno, pa se u tom slučaju računa tzv. težinski GDOP ili KGDOP.

U slučaju kada su odabrana točno četiri satelita, odnosno kada je broj mjerenja jednak broju nepoznanica $M = n$, matrica $\boldsymbol{G}$ u jednadžbi (4.11) je kvadratna. Tada se umjesto generaliziranog inverza može koristiti i regularni inverz [53], a jednadžba poprima oblik:

$$\Delta \boldsymbol{p} = \boldsymbol{G}^{-1} \Delta \boldsymbol{\mu} \tag{4.16}$$

S obzirom na funkcije mjerenja (4.14) Jakobijeva matrica $\boldsymbol{G}$ glasi:

$$\boldsymbol{G} = \begin{bmatrix} \frac{x_1 - x}{d_1} & \frac{y_1 - x}{d_1} & \frac{z_1 - z}{d_1} & c \\ \frac{x_2 - x}{d_2} & \frac{y_2 - x}{d_2} & \frac{z_2 - z}{d_2} & c \\ \frac{x_3 - x}{d_3} & \frac{y_3 - x}{d_3} & \frac{z_3 - z}{d_3} & c \\ \frac{x_4 - x}{d_4} & \frac{y_4 - x}{d_4} & \frac{z_4 - z}{d_4} & c \end{bmatrix}. \tag{4.17}$$

Kako bi smo procijenili pogrešku estimacije potrebno je izračunati odgovarajuću matricu kovarijanci $\mathrm{Cov}(\boldsymbol{p}) = \mathrm{E}\left([\hat{\boldsymbol{p}} - \mathrm{E}(\hat{\boldsymbol{p}})]\,[\hat{\boldsymbol{p}} - \mathrm{E}(\hat{\boldsymbol{p}})]^T\right)$ pri čemu $\mathrm{E}[\cdot]$ predstavlja očekivanu vrijednost. Raspisivanjem izraza (4.10) za $\hat{\boldsymbol{p}}$ koristeći (4.1) i (4.4) dobiva se:

$$\begin{aligned}\hat{\boldsymbol{p}} &= \boldsymbol{p} + (\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\left[\boldsymbol{\mu} - \widetilde{\boldsymbol{f}}(\boldsymbol{p})\right] \\ &= \boldsymbol{p} + (\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\left[\boldsymbol{f}(\boldsymbol{p}) + \boldsymbol{e} - \widetilde{\boldsymbol{f}}(\boldsymbol{p})\right] \\ &= \boldsymbol{p} + (\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\left[\boldsymbol{f}(\boldsymbol{p}) - \boldsymbol{f}(\boldsymbol{p}_0) - \boldsymbol{G}(\boldsymbol{p} - \boldsymbol{p}_0) + \boldsymbol{e}\right]\end{aligned} \tag{4.18}$$

Kako je u gornjem izrazu jedina slučajna varijabla pogreška mjerenja $\boldsymbol{e}$, razlika estimacije i očekivane vrijednosti za estimaciju je:

$$\hat{\boldsymbol{p}} - \mathrm{E}(\hat{\boldsymbol{p}}) = (\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{e} - \mathrm{E}(\boldsymbol{e})) \tag{4.19}$$

te se kovarijanca može izraziti kao:

$$\begin{aligned}\mathrm{Cov}(\boldsymbol{p}) &= \mathrm{E}\left(\left[(\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{e} - \mathrm{E}(\boldsymbol{e}))\right]\left[(\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{e} - \mathrm{E}(\boldsymbol{e}))\right]^T\right) \\ &= (\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\mathrm{E}\left([\boldsymbol{e} - \mathrm{E}(\boldsymbol{e})][\boldsymbol{e} - \mathrm{E}(\boldsymbol{e})]^T\right)\left[\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\right]^T\left[(\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1}\right]^T \\ &= (\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}\boldsymbol{G}(\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1} \\ &= (\boldsymbol{G}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{G})^{-1}\end{aligned} \tag{4.20}$$

pri čemu se koristi definicija za matricu kovarijanci mjerenja $\boldsymbol{\Sigma} = \mathrm{E}\left([\boldsymbol{e} - \mathrm{E}(\boldsymbol{e})][\boldsymbol{e} - \mathrm{E}(\boldsymbol{e})]^T\right)$.

Samo mjerenje vremena dolaska signala podložno je pogrešci te se, kao što je već navedeno, može modelirati kao slučajna varijabla s normalnom razdiobom. Ako se pretpostave jednake standardne devijacije za svaki satelit, tada je matrica kovarijanci mjerenja dijagonalna s jednakim vrijednostima σ_μ^2 na dijagonali, a matrica kovarijanci estimacije vektora $\boldsymbol{p}$ poprima oblik:

$$\mathrm{Cov}(\boldsymbol{p}) = \sigma_\mu^2(\boldsymbol{G}^T\boldsymbol{G})^{-1}. \tag{4.21}$$

što se za GPS može prikazati kao:

$$\mathrm{Cov}(\boldsymbol{p}) = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} & \sigma_{xb} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} & \sigma_{yb} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 & \sigma_{zb} \\ \sigma_{bx} & \sigma_{by} & \sigma_{bz} & \sigma_b^2 \end{bmatrix}. \tag{4.22}$$

Koristeći definiciju koja je dana u jednadžbi (4.12), GDOP se može izraziti preko traga matrice kovarijanci:

$$\begin{aligned} \text{GDOP} &= \frac{\sigma_p}{\sigma_\mu} \\ &= \sqrt{\frac{\text{tr}\left(\text{Cov}(\boldsymbol{p})\right)}{\sigma_\mu^2}} \\ &= \sqrt{\text{tr}\left((\boldsymbol{G}^T\boldsymbol{G})^{-1}\right)}. \end{aligned} \tag{4.23}$$

Kako bi GDOP za GPS bio praktično upotrebljiv, potrebno je dodatno transformirati pogreške u geocentričnom koordinatnom sustavu (x, y, z) u topocentrični (n, e, h) [54]. Ako se prijamnik nalazi na geografskoj širini φ i dužini λ tada se transformacija pogrešaka može obaviti sljedećim izrazom:

$$\begin{bmatrix} \Delta n \\ \Delta e \\ \Delta h \end{bmatrix} = \begin{bmatrix} -\sin\varphi\cos\lambda & -\sin\varphi\sin\lambda & \cos\varphi \\ -\sin\lambda & \cos\lambda & 0 \\ \cos\varphi\cos\lambda & \cos\varphi\sin\lambda & \sin\varphi \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} \tag{4.24}$$

Pretpostavka funkcioniranja ovog načina procjene pogreške lociranja je poznavanje točnih lokacija čvorova koji sudjeluju u estimaciji. To trenutno ograničava upotrebu na slučajeve u kojima se estimira lokacija jednog čvora od strane više čvorova sidara, odnosno čvorova čija lokacija je poznata kao što je gore spomenuti GPS.

4.2.2 Geometrijsko slabljenje preciznosti za sustave s mjerenjem azimuta

Slično kao i za prethodno opisane sustave u kojima se mjeri udaljenost, i u sustavima u kojima se mjeri azimut pogreška lokacije ovisi o pogrešci mjerenja te o relativnom geometrijskom odnosu entiteta koji sudjeluju u određivanju lokacija. Definicija GDOP-a je ista, ali je izračun drukčiji, pa je s obzirom da se algoritam za određivanje položaja koristi upravo izmjerenim azimutima ovaj slučaj u narednom tekstu obrađen detaljnije.

Pod pojmom azimut smatra se kut θ pod kojim sidro vidi čvor čiju lokaciju treba estimirati u odnosu na referentni smjer koji je jednak i poznat svim sidrima kao što je na slici 4.1b pozitivan smjer osi x. Analiza za trodimenzionalni prostor uključuje, osim azimuta, i kut elevacije no kako u većini slučajeva čvorovi leže na istoj ravnini u nastavku se obrađuje procjena točnosti estimacije lokacija za dvodimenzionalni prostor. Funkcija mjerenja azimuta za slučaj kada je referentni smjer jednak pozitivnom smjeru osi x je

dana sljedećom jednadžbom:

$$f_i(\boldsymbol{p}) = \arctan\left(\frac{y-y_i}{x-x_i}\right), i = 1,2,\ldots,M \tag{4.25}$$

Parcijalna derivacija funkcije mjerenja po x osi u točki $\boldsymbol{p}_0 = [x_0, y_0]$ iznosi:

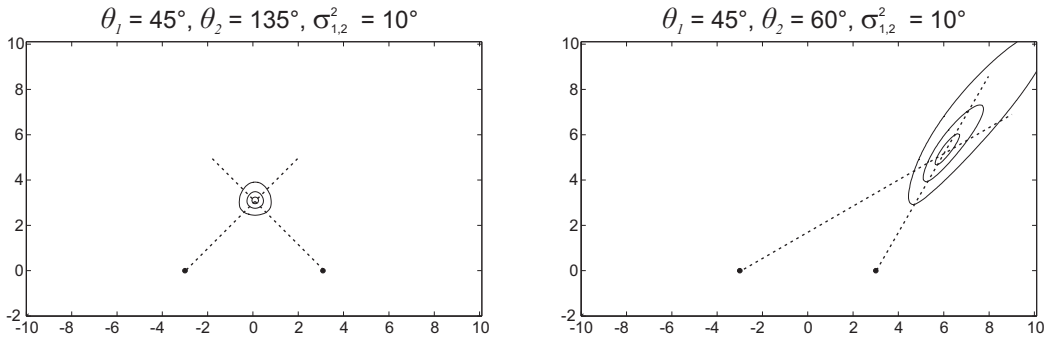
$$\begin{aligned}\left.\frac{\partial f_i}{\partial x}\right|_{\boldsymbol{p}=\boldsymbol{p}_0} &= \frac{1}{1+\left(\frac{y_0-y_i}{x_0-x_i}\right)^2}\cdot\frac{-(y_0-y_i)}{(x_0-x_i)^2} \\ &= \frac{-(y_0-y_i)}{(x_0-x_i)^2+(y_0-y_i)^2} \\ &= -\frac{y_0-y_i}{d_i^2}\end{aligned} \tag{4.26}$$

dok derivacija po y osi ima relativno slično rješenje $\frac{x_0-x_i}{d_i^2}$. Jakobijeva matrica $\boldsymbol{G}$ poprima sljedeći oblik:

$$\boldsymbol{G} = \begin{bmatrix} -\frac{y-y_1}{d_1^2} & \frac{x-x_1}{d_1^2} \\ -\frac{y-y_2}{d_2^2} & \frac{x-x_2}{d_2^2} \\ \vdots & \vdots \\ -\frac{y-y_M}{d_M^2} & \frac{x-x_M}{d_M^2} \end{bmatrix}. \tag{4.27}$$

Daljnji izračun je jednak kao i za slučaj s izmjerenim udaljenostima.

Na slici 4.4 je prikazan utjecaj relativnog geometrijskog položaja na veličinu područja u kojem se nalazi određeni postotak estimacija prema 3σ pravilu (68%-95%-99.7%).



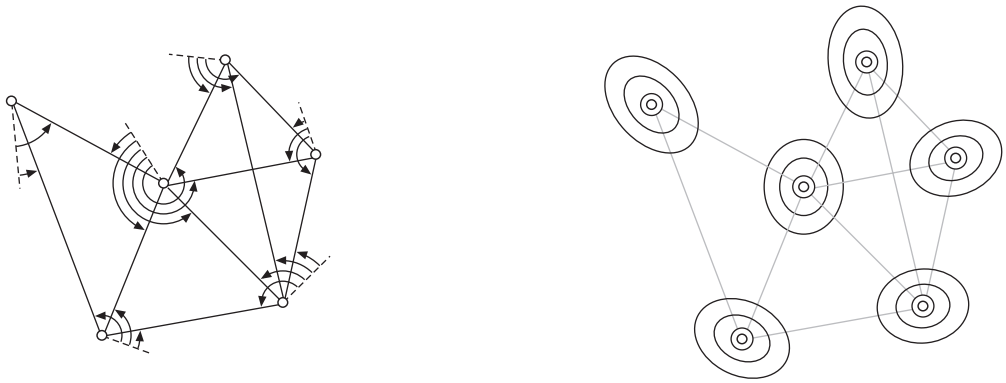
Slika 4.4: Konture funkcije vjerodostojnosti za dva različita položaja čvora čija lokacija se estimira. I u jednom i u drugom slučaju varijanca mjerenja azimuta je $\sigma^2 = 10°$, a sidra su smještena na koordinatama $(-3,0)$ i $(3,0)$.

Zanimljivo je primijetiti kako krivulje jednake gustoće vjerojatnosti nisu elipse zbog nelinearne funkcije estimacije (4.11) koja povezuje izmjerene vrijednosti u formi slučajnih varijabli s normalnom razdiobom, npr. (θ_1,θ_2), i estimiranu lokaciju čvora $(\hat{x},\hat{y})$.

4.3 Procjena točnosti estimacije položaja za neusidrene mreže

Kao što je već spomenuto procijenjena točnost estimiranih lokacija u algoritmima za raspodijeljeno određivanje položaja u neusidrenim mrežama može predstavljati vrlo koristan podatak. U prethodnom dijelu navedena je detaljna analiza za slučaj kada se estimira lokacija jednog čvora koristeći točne lokacije nekoliko čvorova. Procjena točnosti estimiranih lokacija za mreže u kojima ne postoje čvorovi koji poznaju svoju lokaciju predstavlja specifičan slučaj zato jer se sve lokacije estimiraju u relativnom koordinatnom sustavu. Naime, kod estimacije u relativnom koordinatnom sustavu, čak i ukoliko je formacija globalno kruta, postoji beskonačan broj jednakovrijednih rješenja, odnosno kako je spomenuto u dijelu 2.2, takozvanih međusobno kongruentnih formacija.

Čak i pod pretpostavkom poznavanja točnih lokacija, procjena točnosti uključuje transformaciju relativnog u apsolutni koordinatni sustav, odnosno rotaciju, skaliranje i translatiranje, na način da je zbroj kvadrata udaljenosti čvorova u relativnom i apsolutnom koordinatnom sustavu minimalan. Promatrajući svaki čvor posebno uvijek postoji transformacija koja će lokaciju pojedinog čvora iz relativnog koordinatnog sustava prebaciti u apsolutni na takav način da se udaljenost svede na nulu. Stoga prilikom procjene točnosti estimacije u relativnom koordinatnom sustavu jedini valjani kriterij je kumulativna pogreška za sve čvorove formacije kao što je prikazano na slici 4.5.



(a) (b)

Slika 4.5: Procjena točnosti estimacije za neusidrenu mrežu korištenjem izmjerenih azimuta (a) u formi $n\sigma$ elipsa dobivenih iz matrice kovarijanci (b).

U narednom dijelu izvodi se metoda procjene točnosti estimacije za neusidrene mreže s mjerenjem azimuta koristeći faktor geometrijskog slabljenja prema njegovoj definiciji. Definicija je dana u jednadžbi (4.12), a u narednim potpoglavljima izvodi se njen brojnik i nazivnik. Dobiveni izraz nazvan je RGDOP gdje prefiks R označava da se radi o estimaciji lokacije u relativnom koordinatnom sustavu.

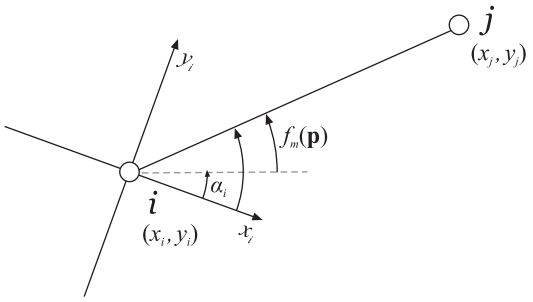
4.3.1 Varijanca estimacije položaja za algoritme s mjerenjem azimuta

Zadani problem određivanja položaja u neusidrenim mrežama korištenjem izmjerenih azimuta glasio bi: *koristeći se* $M \times 1$ *mjernim vektorom* $\boldsymbol{\mu}$ *azimuta* θ_{ij} *između svih parova susjednih čvorova* i *i* j *treba estimirati vektor položaja* $\boldsymbol{p} = \{x_1, y_1, \alpha_1, x_2, y_2, \alpha_2, \ldots, x_n, y_n, \alpha_n\}$, duljine $3n \times 1$.

Algoritam ne pretpostavlja poznavanje globalne orijentacije čvorova, primjerice putem kompasa kojim bi, u tom slučaju, trebao biti opremljen svaki čvor. Stoga se u ovom dijelu obrađuje slučaj estimacije lokacije i orijentacije svakog čvora, odnosno položaja. U tom slučaju jednadžba koja povezuje lokacije čvora i koji obavlja mjerenje i čvora j čiji se azimut određuje transformira se iz (4.25) u:

$$f_m(\boldsymbol{p}) = \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) - \alpha_i,\ m = 1,2,\ldots,M \tag{4.28}$$

pri čemu α_i predstavlja nepoznatu orijentaciju čvora koji obavlja mjerenje prema slici 4.6. Mjerenja su i dalje nezavisna s pretpostavljenom jednakom varijancom σ^2_{AoA} odnosno kovarijancom $\boldsymbol{\Sigma} = \sigma^2_{\text{AoA}}\boldsymbol{I}$.



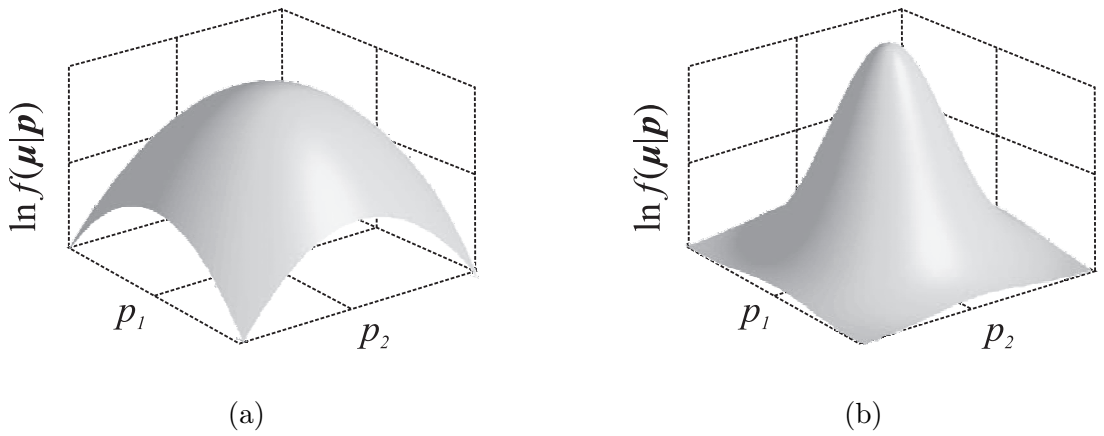
Slika 4.6: Mjerenje azimuta čvora j od strane čvora i pri čemu čvorovi nemaju podatak o svojoj lokalnoj orijentaciji.

Problem određivanja varijance estimacije položaja mjerenjem azimuta u neusidrenim mrežama svodi se na određivanje tzv. Cramér-Rao limita (engl. *Cramér-Rao bound – CRB*) [55, 56]. CRB je donja granična vrijednost matrice kovarijanci bilo kojeg nepristranog estimatora (engl. *unbiased estimator*). Svojstvo CRB-a je da njegova vrijednost ne ovisi o tome koji se algoritam koristi, već on pruža uvid u kvalitetu najbolje moguće estimacije za zadane parametre mreže. Matematički izvod CRB-a za problem određivanja položaja uz pomoć mjerenja udaljenosti dan je u [57]. U [58] je dana procedura za računanje CRB-a za isti problem, ali korištenjem mjerenja vremena dolaska signala TOA, jakosti primljenog signala RSS i smjera dolaska signala AOA.

CRB se izvodi i računa koristeći Fisherovu informacijsku matricu (engl. *Fisher Information Matrix – FIM*) $\mathcal{I}(\boldsymbol{p})$, koja je definirana sljedećim izrazom:

$$\mathcal{I}(\boldsymbol{p}) = -\mathrm{E}\left[\frac{\partial^2}{\partial \boldsymbol{p}^2} \ln f(\boldsymbol{\mu}|\boldsymbol{p})\right]. \tag{4.29}$$

Funkcija $f(\boldsymbol{\mu}|\boldsymbol{p})$ je specijalan slučaj funkcije vjerodostojnosti (4.2) u kojoj je vektoru $\boldsymbol{p}$ dodana orijentacija, a element funkcije $\boldsymbol{f}$ je dan u jednadžbi (4.28). U samom izrazu za FIM prvo se obavlja logaritmiranje čime se riješavamo eksponenta što je moguće jer je eksponencijalna funkcija monotono rastuća. Nadalje, prema estimatoru po maksimalnoj vjerodostojnosti prva derivacija funkcije vjerodostojnosti je prema definiciji jednaka 0, dok je druga derivacija proporcionalna zakrivljenosti krivulje u maksimumu. Kao što je prikazano slikom 4.7, zakrivljenost krivulje funkcije vjerodostojnosti u maksimumu je proporcionalna točnosti same estimacije. U općem slučaju FIM predstavlja količinu informacija koju posjeduje slučajna varijabla $\boldsymbol{\mu}$ o parametru $\boldsymbol{p}$.



(a) (b)

Slika 4.7: Primjer krivulja funkcije vjerodostojnosti s (a) malom i (b) velikom zakrivljenošću [58].

U slučaju kada je distribucija slučajne varijable koja opisuje mjerenja normalna s kovarijancom $\boldsymbol{\Sigma} = \sigma^2_{\mathrm{AoA}}\boldsymbol{I}$, izraz za FIM dan je sljedećom jednadžbom [57]:

$$\mathcal{I}(\boldsymbol{p}) = \frac{1}{\sigma^2_{\mathrm{AoA}}}\boldsymbol{G}^T\boldsymbol{G} \tag{4.30}$$

pri čemu je $\boldsymbol{G}$ već spomenuta Jakobijeva matrica funkcije mjerenja (4.5) veličine $M \times 3n$.

S obzirom na funkciju mjerenja (4.28) njen mn-ti element je jednak:

$$G_{mn} = \frac{\partial \mu_m}{\partial p_n} = \begin{cases} 0, & \text{ako je } p_n \notin \{x_i, y_i, x_j, y_j, \alpha_i\} \\ \frac{y_j - y_i}{d_{ij}^2}, & \text{ako je } p_n = x_i \\ \frac{y_i - y_j}{d_{ij}^2}, & \text{ako je } p_n = x_j \\ \frac{x_i - x_j}{d_{ij}^2}, & \text{ako je } p_n = y_i \\ \frac{x_j - x_i}{d_{ij}^2}, & \text{ako je } p_n = y_j \\ 1, & \text{ako je } p_n = \alpha_i. \end{cases} \tag{4.31}$$

$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ je udaljenost između *točnih* lokacija čvorova i i j.

CRB se definira kao inverz Fisherove informacijske matrice i predstavlja donju graničnu vrijednost za matricu kovarijanci. Ako se CRB izračuna koristeći jednadžbu (4.30):

$$\text{CRB} = \mathcal{I}^{-1} = \sigma_{\text{AoA}}^2 (\boldsymbol{G}^T \boldsymbol{G})^{-1} \tag{4.32}$$

dobiva se upravo izraz (4.21) izveden za matricu kovarijanci kod usidrenih mreža.

FIM se može računati koristeći (4.30), ali pri računanju njenog inverza – matrice CRB, za slučaj neusidrenih mreža nailazi se na problem. Kao što se tvrdi u [59], FIM u slučaju problema lokalizacije neusidrene mreže je nedovoljnog ranga. U [60] dan je dokaz sljedećeg teorema.

Teorem 3. *Za problem lokalizacije neusidrene mreže (korištenjem mjerenja udaljenosti), s ukupno n čvorova, rang Fisherove informacijske matrice $I(P)$ je $2n - 3$.*

Uzrok smanjenom rangu je posljedica ukupno 3 stupnja slobode koje posjeduje rješenje u relativnom koordinatnom sustavu, s obzirom na rotacije i translacije valjanog rješenja. U mrežama u kojima se mjeri azimut, rang FIM-a iznosi $3n - 4$ s obzirom da postoji još jedan parametar po čvoru za estimaciju (orijentacija) i još jedan stupanj slobode (skaliranje). U jednom i u drugom slučaju, FIM je singularna matrica i njen inverz I^{-1} ne postoji. Unatoč tome, ista ta matrica je punog ranga, ako se promatra u potprostoru razapetom s $3n - 4$ ortonormirana svojstvena vektora $\vec{v}_1, \ldots, \vec{v}_{3n-4}$, koji odgovaraju svojstvenim vrijednostima matrice $\mathcal{I}$ različitim od nule, kao što je prezentirano u [60]. Tada je njen inverz dan s $Q^{-1} = V^T \mathcal{I}^\dagger V$, gdje je $V = (v_1, \ldots, v_{3n-4})$, a $\mathcal{I}^\dagger$ je Moore-Penroseov pseudoinverz matrice $\mathcal{I}$.

Već je spomenuto da se kod određivanja lokacija u neusidrenim mrežama ne može limitirati točnost estimacije pojedinog čvora, jer bi se estimirano rješenje uvijek moglo translatirati tako da se poklopi s pravom vrijednosti. Prema tome, jedini relevantan limit estimacije mora biti invarijantan na transformacije, a to je ukupni limit estimacije

označen kao $V_{\text{ukupno}} = \text{tr}(\mathcal{I}^{\dagger})$.

Za izračun faktora geometrijskog slabljenja preciznosti treba izračunati korijen srednje kvadratne pogreške lokacije (engl. *root mean square error – RMSE*) koji je definiran s:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left((\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)\right)^2} \tag{4.33}$$

U tu svrhu se iz matrice kovarijanci odnosno CRB-a izostavlja orijentacijski dio te se izdvajaju elementi na dijagonali koji odgovaraju varijancama koordinata x i y:

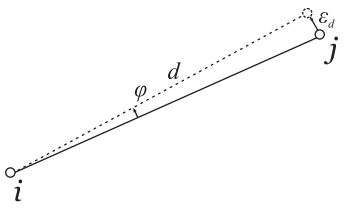
$$\sigma_{p_{\text{CRB}}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\text{E}\left((\hat{x}_i - x_i)^2\right) + \text{E}\left((\hat{y}_i - y_i)^2\right)\right)} \tag{4.34}$$

$$= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{I}^{\dagger}_{3i-2} + \mathcal{I}^{\dagger}_{3i-1}\right)}. \tag{4.35}$$

gdje je $(\hat{x}_i,\hat{y}_i)$ estimirana lokacija čvora i.

4.3.2 Varijanca mjerenja udaljenosti

Kako bi smo postigli željenu neovisnost faktora geometrijskog slabljenja preciznosti o varijanci mjerenja azimuta, potrebno je tu varijancu izraziti kao funkciju varijance mjerenja udaljenosti. Situacija koju analiziramo prikazana je na slici 4.8.



Slika 4.8: Pogreška mjerenja azimuta φ u odnosu na odgovarajuću pogrešku mjerenja udaljenosti ε_d.

Ako se pretpostavi dovoljno mala pogreška mjerenja azimuta vrijedi:

$$\varphi \approx \frac{\varepsilon_d}{d}. \tag{4.36}$$

U tom slučaju srednja kvadratna pogreška mjerenja udaljenosti koja proizlazi iz pogreške mjerenja azimuta [9] iznosi:

$$\sigma_d^2 = \frac{1}{M}\sum_{i=1}^{M} d_i^2 \sigma^2_{\text{AoA}i} \tag{4.37}$$

4.4 RGDOP

Konačno, izraz za faktor geometrijskog slabljenja preciznosti u neusidrenim mrežama može se izvesti korištenjem izraza (4.35) i (4.37):

$$\begin{aligned} \mathrm{RGDOP} &= \frac{\sigma_{p_{\mathrm{CRB}}}}{\sigma_d} \\ &= \frac{\sigma_{\mathrm{AoA}} \sqrt{\frac{1}{n} \operatorname{tr}\left(\boldsymbol{G}^T \boldsymbol{G}\right)^{-1}}}{\sigma_{\mathrm{AoA}} \sqrt{\frac{1}{M} \sum_i^M d_i^2}} \\ &= \sqrt{\frac{M}{n} \frac{\operatorname{tr}\left(\boldsymbol{G}^T \boldsymbol{G}\right)^{-1}}{\sum_i^M d_i^2}} \end{aligned} \tag{4.38}$$

pri čemu se pretpostavlja kako su varijance mjerenja azimuta za sve parove čvorova jednake i dovoljno male da se može koristiti aproksimacija (4.36). Osnovna svojstva RGDOP-a kod estimacije lokacija mjerenjem azimuta su sljedeća:

- ovisi samo o formaciji, odnosno relativnom rasporedu čvorova i pripadajućem grafu
- ne ovisi o apsolutnoj veličini formacije
- ne ovisi o standardnoj devijaciji mjerenja σ_{AoA}.

4.5 Rezultati simulacija

U obrađenoj literaturi u kojoj se analiziraju granice točnosti estimacije položaja za izračun Cramér-Rao limita uvijek se koriste točne lokacije čvorova. Naime, specifični ciljevi tih analiza su analiza kvalitete algoritama za estimaciju položaja usporedbom s donjom granicom koju predstavlja CRB [57, 60], rastav pogreške estimacije kod usidrenih mreža na relativni i transformacijski dio [56], te odabir i procjena kvalitete položaja sidara s obzirom na estimaciju lokacije ostalih čvorova u mreži [61]. Kako se u ovom slučaju analizira određivanje lokacija u neusidrenim mrežama, postavljena je hipoteza kako se za procjenu kvalitete mogu koristiti estimirane lokacije. Za potvrdu hipoteze potrebno je estimairati lokacije u ispitnoj mreži te usporediti srednju kvadratnu pogrešku s estimiranom.

U tu svrhu proveden je eksperiment koji se sastoji od niza simulacija. S obzirom na veliki broj parametara koji je moguće varirati u eksperimentu, odabrane su one relevantne te je određen njihov raspon. U nastavku je dana analiza i konačni odabir parametara eksperimenta.

Broj čvorova i graf mreže Konačni cilj je koristiti procjenu kvalitete estimacije u algoritmu kao parametar prema kojem će se filtrirati formacije nakon inicijalnog određivanja položaja. Kako bi smo izbjegli pojavu formacija koje nisu globalno

krute te pojavu višestrukih rješenja, jedna od mogućih klasa inicijalnih formacija koje se koriste su najveći segmenti grafa mreže koji čine potpuni graf, odnosno takozvane klike (engl. *clique*). S obzirom na prosječni broj susjeda koji u tipičnoj mreži iznosi između 5 i 15, veličine inicijalnih klika koje se očekuju su između 3 i 7.

Izbor reprezentativnih formacija Kako bi se mogla odrediti korelacija između točnih i procijenjenih pogrešaka estimacije, za svaku odabranu veličinu mreže generirano je 1000 slučajnih formacija. Nakon toga je između njih odabran podskup od 10 reprezentativnih. Kriterij po kome su odabrane reprezentativne je iznos GDOP-a tako da je kod 10 izabranih mreža on jednoliko raspoređen između najmanje i najveće zabilježene vrijednosti unutar inicijalnih 1000 mreža.

Standardna devijacija mjerenja S obzirom na jedan od postavljenih kriterija korištenja RGDOP-a, pogreške mjerenja bi trebale biti relativno male, kao što su i očekivane male vrijednosti za osjetilo obrađeno u eksperimentalnom djelu istraživanja. Stoga se simuliraju slučajevi u kojima je standardna devijacija mjerenja azimuta između 1° i 5°.

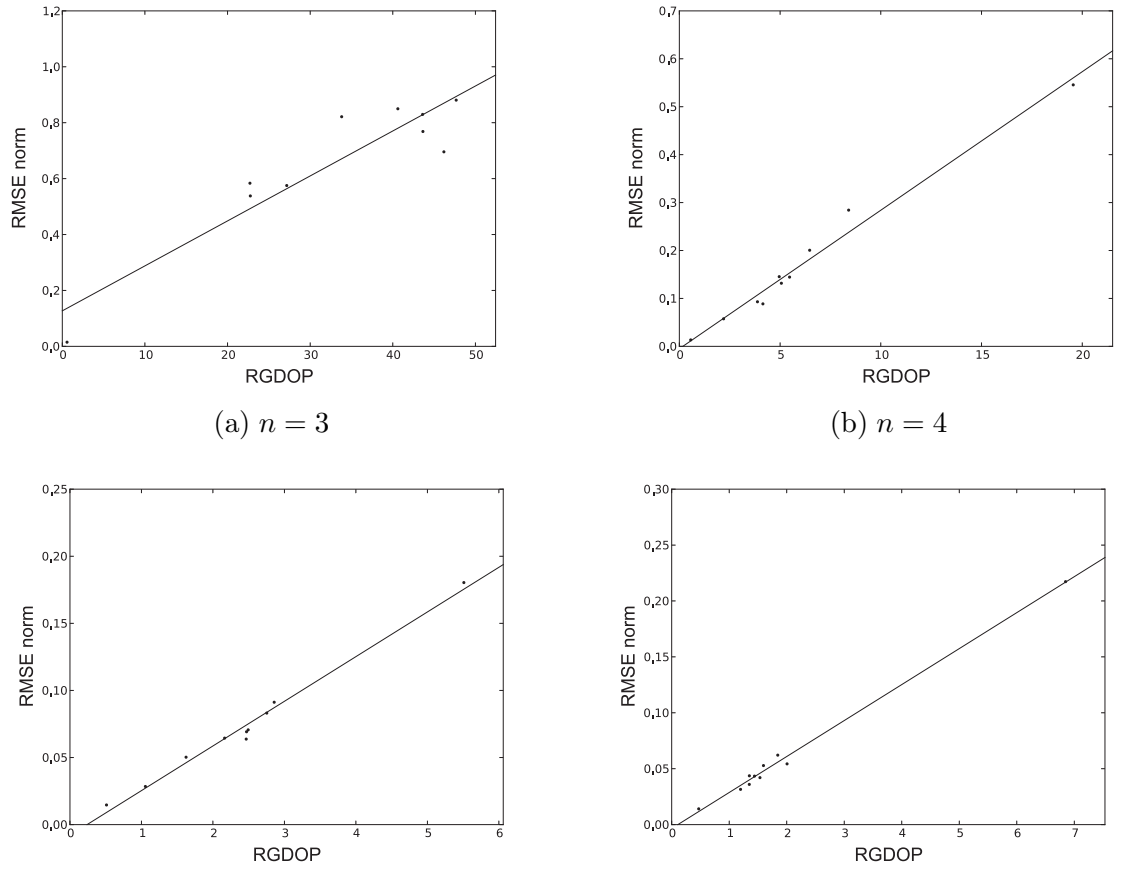
RGDOP ne ovisi o apsolutnoj veličini formacije, a srednja kvadratna pogreška ovisi, pa se u svrhu usporedbe pogreška normira po veličini. Normiranje se obavlja tako da se korijen srednje kvadratne pogreške (jednadžba (4.33)) podijeli normom formacije. Norma formacije se dobiva translacijom formacije tako da njena centroida bude u ishodištu koordinatnog sustava te se zatim koristi sljedeća jednadžba:

$$(G,p)_{\text{norm}} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{x_i^2 + y_i^2}. \tag{4.39}$$

Dijeljenjem srednje kvadratne pogreške normom ona se svodi na bezdimenzionalnu veličinu koja se može usporediti s RGDOP-om.

Rezultati eksperimenta kada je mjerenje iznimno točno, odnosno za standardnu devijaciju mjerenja $\sigma_{\text{AoA}} = 1°$ prikazani su na slici 4.9. Iz dobivenih rezultata može se zaključiti kako je hipoteza potvrđena, te kako se RGDOP izračunat iz estimiranih lokacija može koristiti za procjenu točnosti estimacije lokacija ukoliko su mjerenja uz pomoć kojih se estimiraju lokacije dovoljno točna, u ovom slučaju $\sigma_{\text{AoA}} = 1°$.

Međutim, rezultati također pokazuju kako je korelacija između RGDOP-a i normirane pogreške izraženija u slučajevima kada se formacija sastoji od više čvorova. Kako bi prikazali ovu ovisnost, na slici 4.10 su prikazane vrijednosti Paersonovog koeficijenta korelacije za svaku kombinaciju preciznosti mjerenja i broja čvorova u formaciji. S obzirom na rezultate, zaključak je kako je utjecaj broja čvorova na korelaciju veći od utjecaja same preciznosti mjerenja kako je prvobitno bilo pretpostavljeno. Razlog tome je kvadratna ovisnost ($n(n-1)$) između broja čvorova i mjerenja u formacijama s potpunim



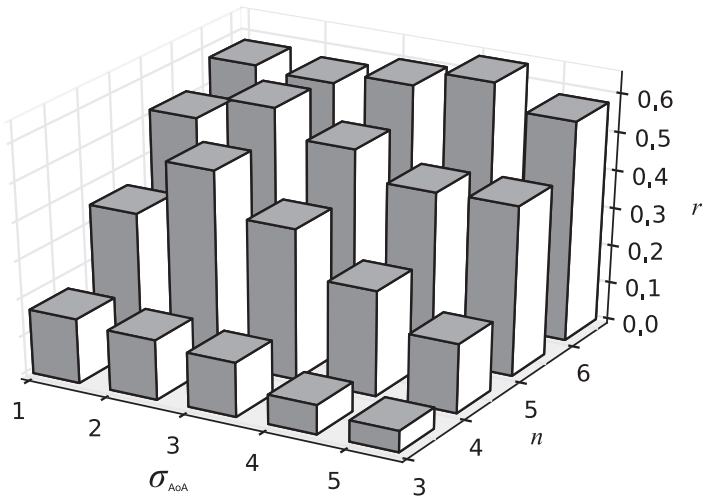
(a) $n = 3$ (b) $n = 4$

(c) $n = 5$ (d) $n = 6$

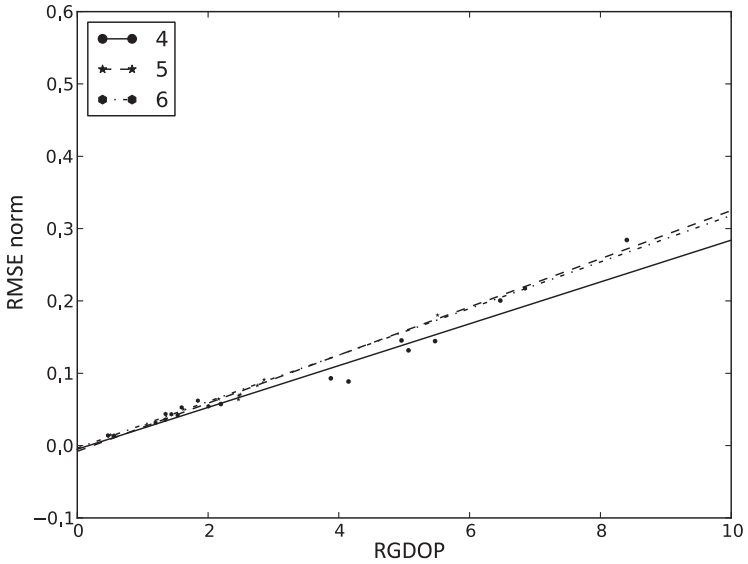
Slika 4.9: Odnos između faktora geometrijskog slabljenja preciznosti za neusidrene mreže (RGDOP) dobivenog iz estimiranih lokacija i njihove normirane srednje kvadratne pogreške uz standardnu devijaciju mjerenja $\sigma_{\mathrm{AoA}} = 1°$ i broj čvorova od 3 do 6.

grafom. Zbog toga se uz istu preciznost mjerenja dobivaju preciznija estimacija lokacije, a time i RGDOP-a što vodi do bolje korelacije.

Zanimljivo je primijetiti kako postoji veliki skok u vrijednosti koeficijenta korelacije između formacija s 3 i sa 4 čvora. Na slici 4.11 prikazano je kako za formacije koje sadrže više od 4 čvora pravac regresije ima sličan nagib. Ta činjenica omogućava algoritmu korištenje generičkog praga prema kojem može donositi odluku o korištenju rezultata estimacije lokacija pojedinih formacija na način kao što će biti prikazano u poglavlju 5.



Slika 4.10: Pearsonov koeficijent korelacije r za različite vrijednosti standardne devijacije mjerenja σ_{AoA} i broj čvorova u formaciji n.



Slika 4.11: Odnos RGDOP-RMSE za $\sigma_{\mathrm{AoA}} = 1^\circ$ za formacije od 4 do 6 čvorova imaju skoro identične pravce regresije što se može iskoristiti u algoritmu za određivanje praga koji ne ovisi o broju čvorova formacije.

Poglavlje 5

Algoritam za određivanje položaja

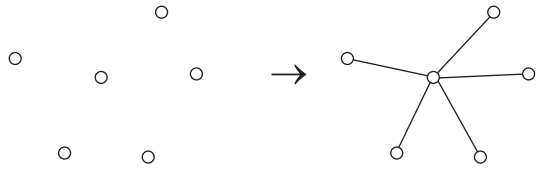
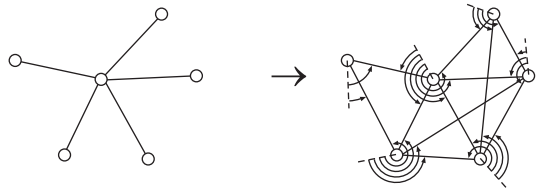
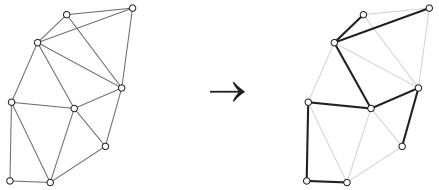
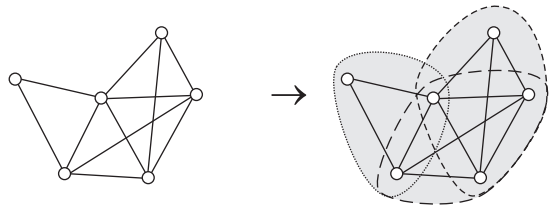
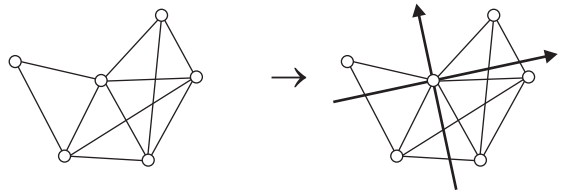
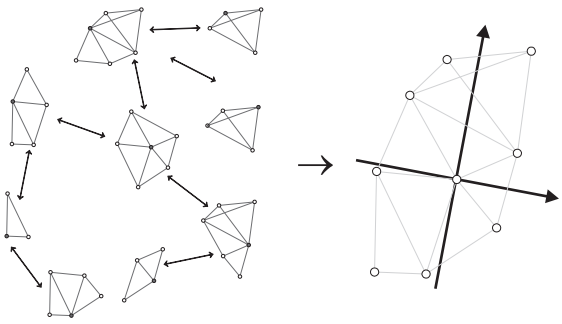
Osnovni cilj algoritma za određivanje položaja je što točnije odrediti lokaciju i orijentaciju čvorova mreže. S obzirom na inherentna energetska i računalna ograničenja čvorova bežične mreže osjetila, njihove dimenzije i cijenu, te posljedično smanjenu pouzdanost, dodatni ciljevi su: minimizacija potrošnje energije, robusnost na prestanak rada pojedinih čvorova, kao i izbjegavanje potrebe za posebno opremljenim čvorovima, poput sidara. Upravo ti dodatni ciljevi uvjetuju odabir tipa algoritma, te prema klasifikacijama spomenutim u 3. poglavlju, izvorni algoritam opisan u nastavku spada u domenu raspodijeljenih algoritama u neusidrenim mrežama.

Generički cilj svakog raspodijeljenog algoritma je povećanje razine znanja entiteta koji sudjeluju u njegovom izvršavanju. Znanje se u tom smislu sastoji od informacija, a konačni cilj algoritma za određivanje položaja je informacija o koordinatama i orijentaciji pojedinog čvora u mreži. Kako bi čvor uspješno došao do te informacije, potrebno je osigurati preduvjete odnosno informacije koje su potrebne za funkcioniranje dijela algoritma u kojem se estimira položaj.

Cjelokupni algoritam sastoji se od dvije faze. Prva je pripremna, u kojoj se izvršavaju algoritmi za otkrivanje čvorova susjeda, formiranje komunikacijskog stabla i odabir vođe, te mjerenje odnosa među čvorovima. Druga uključuje određivanje lokacije u lokalnim koordinatnim sustavima, te spajanje istih kako bi se odredile lokacije u globalnom koordinatnom sustavu. U tablici 5.1 prikazan je niz zadataka koje je potrebno obaviti za uspješnu estimaciju položaja te popis odgovarajućih raspodijeljenih algoritama koji se koriste u tu svrhu. U tablici su navedeni preduvjeti odnosno informacije koje oni zahtijevaju na početku izvršavanja, te informacije koje pružaju nakon izvršenja.

U nastavku se daje formalna definicija mrežnih formacija kako bi se zatim opisali detalji svakog od navedenih zadataka druge faze algoritma. Nakon toga se predstavljaju potencijalne heuristike koje se zasnivaju na rezultatima iz prethodnog poglavlja, te se odabire najpogodnija. U konačnici se daju rezultati provedenih simulacija s naglaskom

Tablica 5.1: Dvije faze algoritma za estimaciju položaja i njihovi zadaci

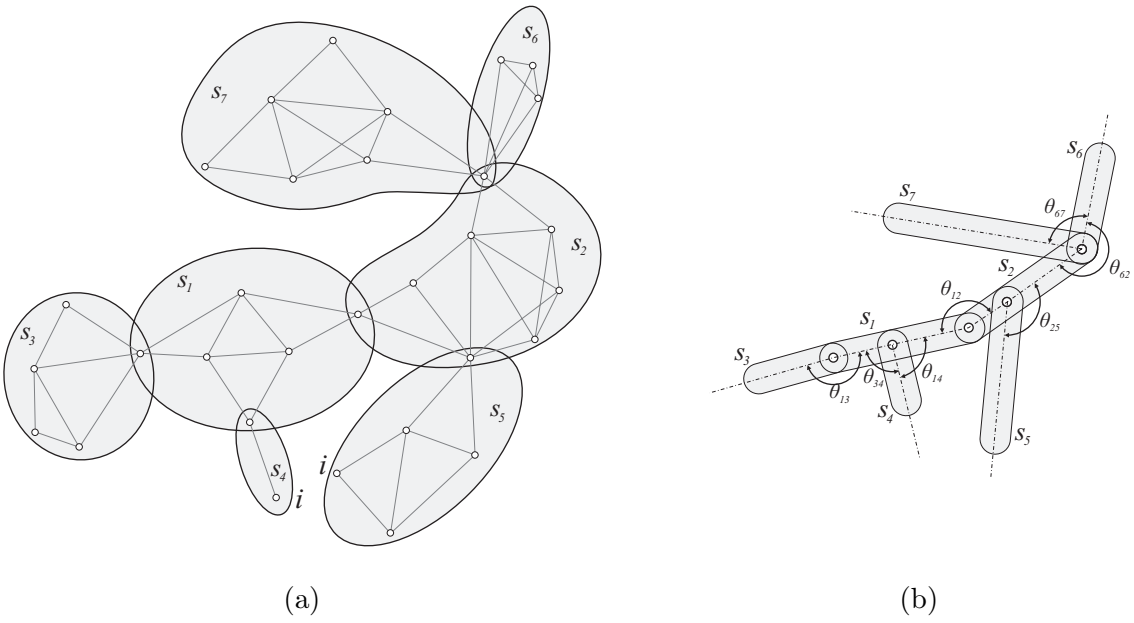
1. Faza: Osiguravanje preduvjeta	
Zadatak: Detekcija susjeda u komunikacijskom grafu **Algoritam:** Wake-up [62] **Preduvjet:** jedinstveni ID broj čvora **Rezultat:** lista čvorova s kojima je moguće ostvariti dvosmjernu komunikaciju	
Zadatak: Mjerenje azimuta i razmjena mjernih podataka, detekcija susjeda u grafu mjerenja[1] **Algoritam:** Flood **Preduvjet:** lista susjeda **Rezultat:** lista čvorova s kojima je moguće obaviti mjerenje azimuta i pripadajuća matrica mjerenja	
Zadatak: Formiranje stabla u grafu mjerenja i izbor vođe (korijena) **Algoritam:** MegaMerger, Broadcast **Preduvjet:** lista susjeda u grafu mjerenja **Rezultat:** čvor roditelj i lista čvorova djece, zajedno čine podskup susjeda u grafu mjerenja	
2. Faza: Određivanje lokacija	
Zadatak: Podjela susjedstva na podgrozdove[2] **Algoritam:** Sc2c, Clique **Preduvjet:** lista susjeda u grafu mjerenja **Rezultat:** lista listi susjeda u grafu mjerenja čvorovi se mogu nalaziti u više listi, a unija svih listi je jednaka skupu svih susjeda	
Zadatak: Određivanje položaja u inicijalnim grozdovima[2] **Algoritam:** RAST, MiaIniLoc, iterativna triangulacija **Preduvjet:** mjerenja azimuta svih susjeda i njihova mjerenja za njihove susjede **Rezultat:** lista listi susjeda u grafu mjerenja, gdje je svakom čvoru pridružen podatak o položaju $[x, y, \alpha]$	
Zadatak: Spajanje grozdova[2] **Algoritam:** BroadcastStitch, ConvergecastStitch **Preduvjet:** lokacije čvorova u grozdu[2], stablo **Rezultat:** lista listi čvorova nasljednika u stablu s podacima o njihovom položaju, posljedično korijen ima podatke o lokacijama svih čvorova	

[1]Iako je na slici graf mjerenja istovjetan komunikacijskom grafu, to ne mora biti uvijek slučaj.
[2]Definicije grozda i podgrozda dane su nastavku poglavlja, u dijelu 5.1.

na usporedbu preciznosti sa i bez korištenja odabrane heuristike te usporedbu s centraliziranom verzijom algoritma za nekonveksne topologije.

5.1 Mrežne strukture

Algoritam za određivanje položaja koristi se metodom *lokaliziraj-i-spoji* te se tijekom njegovog izvršavanja obavlja izračun i spremaju se podaci o položajima čvorova unutar inicijalnih lokalnih koordinatnih sustava, djelomično spojenih koordinatnih sustava i, u konačnici, globalnog koordinatnog sustava. S obzirom na mogućnost pojave višestrukih rješenja, što je spomenuto kod opisa teorije krutosti u dijelu 2.2, svaki od tih sustava može biti dodatno segmentiran. Kako bi se omogućilo ispravno korištenje dobivenih podataka unutar algoritma i kako bi se olakšao opis samog algoritma, u nastavku se daje formalna definicija mrežnih struktura u kojima čvor računa svoju lokaciju, te njihova svojstva. Pod pojmom mrežne strukture, podrazumijevaju se logički dijelovi mreže koji su bazirani na topologiji i svojstvima krutosti.



(a) (b)

Slika 5.1: Grozd podijeljen na podgrozdove s_1 do s_7 (a) pri čemu podgrozdovi mogu biti prikazani kao elementi stabla ograničeni izmjerenim azimutima zajedničkih čvorova θ_{ij} i njihovim lokacijama (b).

Podgrozd je osnovna mrežna struktura. Elementi podgrozda su čvorovi koji čine formaciju koja je paralelno kruta. S obzirom na ovu definiciju, lokacije čvorova koje pripadaju istom grozdu moguće je jedinstveno odrediti. Kako bi se ovo svojstvo održalo tijekom izvršavanja algoritma, sve kongruentne transformacije obavljaju se isključivo na razini podgrozda. U memoriji čvora podgrozd je zapisan u formi

polja odnosno liste. Osnovni element tog polja je ID broj čvora te njegova lokacija i orijentacija.

Grozd je struktura koja se sastoji od više podgrozdova. Grozdovi nastaju prilikom spajanja podgrozdova koji nakon spajanja ne čine paralelno krutu formaciju stoga lokacije čvorova nisu jedinstveno određene. Posljedično lokacije pojedinih čvorova unutar grozda koji se nalaze unutar više podgrozdova ne moraju biti jednake, kao što je to prikazano na slici 5.1a za čvor označen s i koji se nalazi u podgrozdovima s_3 i s_4. S obzirom da su podgrozdovi ispravno rotirani, izmjereni azimuti su međusobno usklađeni. *Inicijalni grozd* je grozd koji se formira u svakom čvoru mreže na samom početku izvršavanja algoritma, a sastoji se od čvora u kojemu je pohranjen i čvorova koji se u grafu mreže nalaze na udaljenosti manjoj od određenog unaprijed zadanog broja. *Globalni grozd* je grozd koji sadrži sve čvorove mreže, njegovi podgrozdovi su u u idealnom slučaju najveće paralelno krute formacije koje je moguće ostvariti što je sam po sebi zahtjevan zadatak detaljno obrađen u [39]. Kako bi stvaranje globalnog grozda bilo moguće, graf mreže mora biti povezan. U memoriji čvora podgrozd je zapisan u formi polja odnosno liste čiji elementi su podgrozdovi.

Daljnjom analizom grozd se može prikazati kao struktura slična stablu koja se sastoji od podgrozdova povezanih zajedničkim čvorovima kao što je prikazano na slici 5.1b.

5.2 Kreiranje inicijalnih grozdova

Zadatak prve faze algoritma za određivanje položaja je kreiranje inicijalnih grozdova u svakom čvoru mreže. Inicijalni grozd se sastoji od čvora i njegovih prvih susjeda, odnosno u ovom slučaju prema definiciji grozda, minimalna udaljenost koja se razmatra je 1. Bitno je naglasiti kako se navedena udaljenost, odnosno susjedstvo, promatra u tzv. mjernom grafu mreže čiji bridovi se nalaze između čvorova koji mogu međusobno izmjeriti azimut, a ne u komunikacijskom grafu. U provedenim simulacijama i analizi se pretpostavlja da su komunikacijski i mjerni graf identični. To se može učiniti jer navedena pretpostavka ne mijenja parametre koji se u simulaciji promatraju i analiziraju. Samo kreiranje inicijalnih grozdova može se razdvojiti na dva neovisna zadatka, a to su segmentacija inicijalnog grafa na podgrozdove i određivanje inicijalnih lokacija u tako dobivenim podgrozdovima.

5.2.1 Segmentacija inicijalnog grafa na podgrozdove

Preduvjet pri rješavanju ovog zadatka je poznavanje inicijalnog grafa odnosno grafa mreže kojem pripadaju čvorovi iz susjedstva. Zadatak se može riješiti na nekoliko različitih načina od kojih se mogu izdvojiti sljedeća dva:

1. *Sc2c* je metoda kojom se u podgrozdove promoviraju najveći podgrafovi inicijalnog grafa koji imaju takozvani bilateralni poredak čvorova (engl. *bilateration ordering*) [63]. Graf ima bilateralan poredak ukoliko njegovi vrhovi mogu biti poredani tako da vrhovi v_1, v_2 i v_3 čine potpuni graf, a svaki sljedeći vrh v_i je incidentan sa barem dva vrha v_j pri čemu je $j < i$.
2. *ScClique* je metoda kojom se u podgrozdove promoviraju najveći potpuno povezani podgrafovi inicijalne formacije, odnosno klike. Prednost ove metode je što se za svaki podgrozd osigurava maksimalni mogući broj mjerenja. U usporedbi s prethodnim pristupom, mana ovog je što algoritam za pronalazak maksimalnih klika u grafu [64] ima složeniju izvedbu te što je NP-potpun, odnosno složenost mu je eksponencijalna. S obzirom na relativno mali očekivani broj čvorova u formaciji, velika složenost ne mora igrati presudnu ulogu u odluci.

5.2.2 Određivanje inicijalnih lokacija u podgrozdovima

Nakon podjele inicijalne formacije u podgrozdove, potrebno je odrediti lokacije čvorova koje su, s obzirom na definiciju podgrozda, jedinstvene. Ovaj zadatak može se obaviti na različite načine, od kojih su neki spomenuti u dijelu 3.2. Za implementaciju u simulatoru odabrana je metoda robusne angulacije korištenjem potprostornih tehnika (engl. *Robust Angulation using Subspace Techniques – RAST*), opisana u [47], a koja spada u metode koje se zasnivaju na višedimenzionalnom skaliranju. Detaljni opis metode i algoritma u inačici koja je implementirana u simulatoru dan je kao prilog P1.

5.3 Spajanje grozdova

Nakon definiranja mrežnih struktura osnovna operacija u procesu spajanja lokalnih koordinatnih sustava je spajanje dva grozda. Ova operacija svodi se na dva zadatka. Prvi zadatak je između svih parova podgrozdova koji se mogu spojiti, odnosno koji imaju dovoljan broj zajedničkih čvorova, odabrati onaj par koji je najbolji prema izabranom kriteriju. Kriterij odabira predstavlja jedan od parametara algoritma, a u implementaciji u simulatoru se koristi broj zajedničkih čvorova između dva podgrozda. Drugi zadatak je spojiti odabrane podgrozdove.

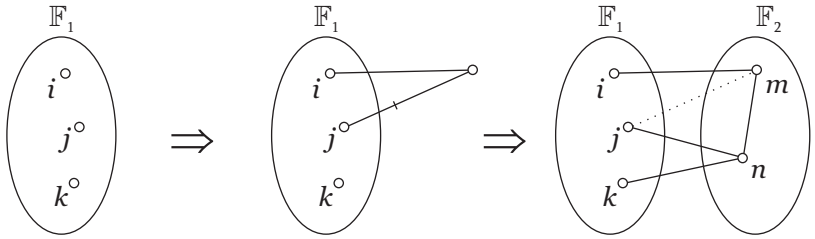
Spajanje podgrozdova podrazumijeva određivanje položaja svih čvorova takozvanog *izvorišnog* podgrozda u koordinatnom sustavu *odredišnog* podgrozda. Minimalni uvjeti koji su potrebni da bi se dva podgrozda mogla spojiti u jedan, na način da se održi osnovno svojstvo podgrozda, a to su jedinstvene lokacije, ovisi o tipu mjerenja. Kod formacija dobivenih mjerenjem udaljenosti minimalan potreban broj zajedničkih čvorova u generičkoj formaciji je 3, a kod mjerenja azimuta 2.

Spajanje dviju krutih formacija s udaljenostima kao ograničenjima, analizirano je i opisano u [65]. Izvornim radom autora [66] ta je analiza proširena i primijenjena na mreže s ograničenjima azimuta u dvodimenzionalnim prostorima. U tom radu definirani su i dokazani sljedeći teoremi.

Teorem 4. *Neka su dvije paralelno krute formacije* $\mathbb{F}_1$ *i* $\mathbb{F}_2$ *povezane skupom bridova* $\mathcal{L}$. *Slijedi da je* $\mathbb{F}_1 \cup \mathbb{F}_2 \cup \mathcal{L}$ *globalno paralelno kruta ako su zadovoljena sljedeća dva uvjeta:*

1. *Barem tri vrha incidentna bridovima iz skupa* $\mathcal{L}$ *nalaze se u formaciji* $\mathbb{F}_1$ *i barem dva se nalaze u formaciji* $\mathbb{F}_2$.
2. $\mathcal{L}$ *sadrži barem tri brida.*

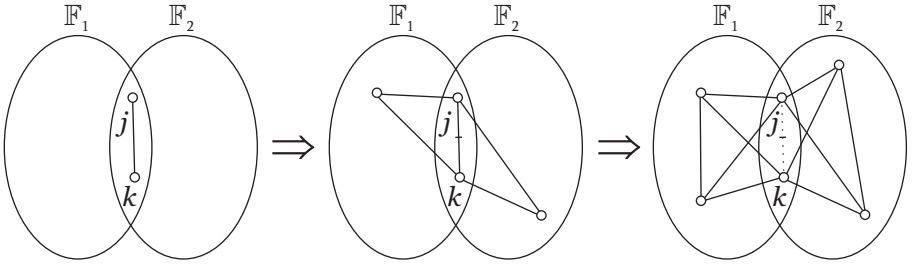
Dokaz. Izaberu se tri vrha i,j,k iz formacije $\mathbb{F}_1$. Koristeći slijed koji se sastoji od jednog dodavanja vrha i jednog dijeljenja brida, ta tri vrha mogu se povezati s vrhovima m,n u formaciji $\mathbb{F}_2$ koristeći tri brida. Slika 5.2 prikazuje ove dvije operacije, a obzirom da dijeljenje brida i dodavanje vrha čuvaju svojstvo paralelne krutosti formacije, kako je pokazano u [67], rezultirajuća formacija je paralelno kruta. Slijedi da se može, počevši od dva vrha m,n, konstruirati proizvoljnu paralelno krutu formaciju $\mathbb{F}_2$ bez da se mijenja formacija $\mathbb{F}_1$ i tri umetnuta brida. □



Slika 5.2: Teorem 4 – dodavanje vrha i djeljenje brida.

Teorem 5. *Ako dvije paralelno krute formacije* $\mathbb{F}_1$ *i* $\mathbb{F}_2$ *dijele najmanje dva vrha, formacija* $\mathbb{F}_1 \cup \mathbb{F}_2$ *je paralelno kruta.*

Dokaz. Ako su j i k vrhovi zajednički formacijama $\mathbb{F}_1$ i $\mathbb{F}_2$ tada se može dodati implicitno ograničenje azimuta između ta dva vrha u obliku brida (j,k) i to učiniti u obje paralelno krute formacije $\mathbb{F}_1$ i $\mathbb{F}_2$ koje sada dijele taj novi brid. Počevši od brida (j,k) slijedom operacija dodavanja vrhova i dijeljenja bridova koje čuvaju paralelnu krutost, može se, nezavisno, unutar svake od formacija, konstruirati proizvoljne paralelno krute formacije koje će dijeliti vrhove j i k i uvijek će se zadržati svojstvo da je formacija $\mathbb{F}_1 \cup \mathbb{F}_2$ paralelno kruta. □



Slika 5.3: Teorem 5 – dodavanje vrha i dijeljenje zajedničkog brida (j,k).

S obzirom na dominantnu frekvenciju pojavljivanja drugog slučaja, u simulatoru je implementiran samo slučaj naveden u teoremu 5.

Kako bi se obavilo spajanje, potrebno je estimirati parametre kongruentnih transformacija, odnosno odrediti faktor skaliranja ξ, vektor translacije $\vec{t}$ i matricu rotacije $\boldsymbol{R}$ te uz pomoć jednadžbe (5.1) lokacije čvorova u izvorišnom koordinatnom sustavu p_s transformirati u lokacije u odredišnom koordinatnom sustavu p_d.

$$p_d = \vec{t} + \xi \mathbf{R} p_s. \tag{5.1}$$

Parametri se estimiraju optimizacijom prema sljedećem kriteriju:

$$\min_{\xi,\vec{t},\boldsymbol{R}} \sum_{i}^{n_z} \|p_{d_i} - \vec{t} - \xi \boldsymbol{R} p_{s_i}\|^2 \tag{5.2}$$

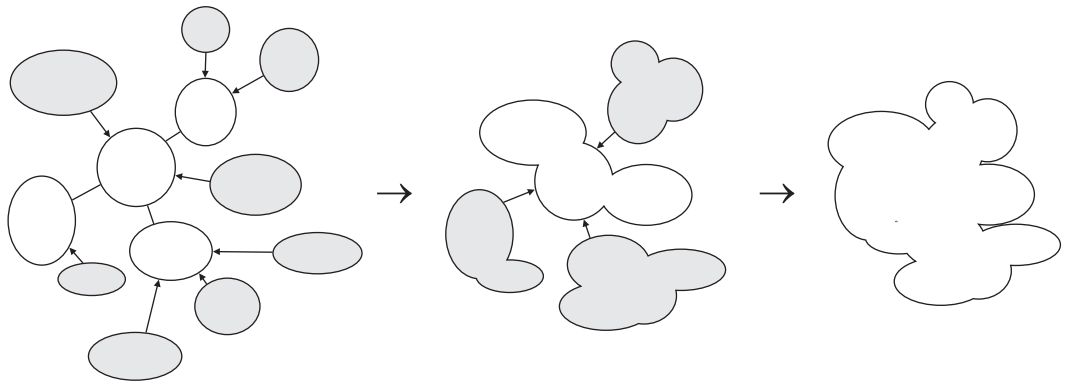
pri čemu n_z označava broj zajedničkih čvorova, a p_{s_i} i p_{d_i} njihove lokacije. Drugim riječima, minimizira se suma kvadrata udaljenosti između lokacija zajedničkih čvorova u odredišnom koordinatnom sustavu i lokacija dobivenih transformacijom iz izvorišnog koordinatnog sustava. Analitičko rješenje za ovaj problem, predloženo u [68], integrirano je u algoritam implementiran u simulatoru, a detaljan opis rješenja dan je u prilogu P2.

5.3.1 Redoslijed spajanja grozdova

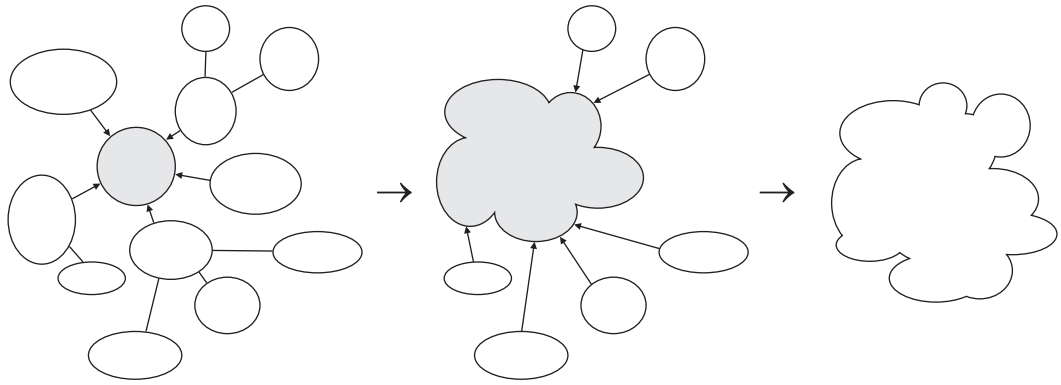
Spajanje grozdova na lokalnoj razini obavlja se izravno, komunikacijom između dva susjedna čvora. U [49] autori analiziraju utjecaj različitih puteva spajanja i zaključuju kako je, s aspekta točnosti, optimalna metoda donošenja odluka o poretku spajanja ona koja funkcionira na razini cijele mreže, sa svim dostupnim podacima, drugim riječima centralizirana. S druge strane, ako se promatra komunikacijsko opterećenje, odluke je puno bolje donositi na lokalnoj razini što se, s obzirom na željenu raspodijeljenu prirodu algoritma, pokušava ostvariti i u ovom algoritmu. Prilikom određivanja redoslijeda spajanja odvojeno se mogu razmatrati tri zadatka:

- odabir bridova duž kojih se obavlja spajanje
- odabir globalnog redoslijeda odnosno smjera spajanja
- odabir lokalnog redoslijeda.

S obzirom da je potrebno spojiti sve grozdove u mreži i to tako da se minimizira ukupan broj spajanja, minimalni broj spajanja potreban da se objedini cijela mreža iznosi $n-1$, pri čemu je n broj inicijalnih grozdova. U slučaju kada se inicijalni grozdovi sastoje od prvih susjeda, n označava također i broj čvorova. Struktura koja odgovara postavljenim zahtjevima minimalnog broja bridova koji povezuju sve vrhove grafa je stablo. Različiti putovi spajanja u vidu cijena bridova minimalnog razapinjajućeg stabla analizirani su u [66] gdje se pokazalo kako ispitani kriteriji (broj susjeda i visina stabla) nisu utjecali na kvalitetu estimiranih lokacija.



(a) *ConvergecastStitch*



(b) *BroadcastStitch*

Slika 5.4: Globalni smjer spajanja grozdova u stablu (a) od listova prema korijenu i (b) od korijena prema listovima

Dva izbora za globalni redoslijed odnosno smjer spajanja u stablu spajanja su (1) od listova prema korijenu koji se, prema odgovarajućem algoritmu Convergecast, naziva *ConvergecastStitch*, te (2) od korijena prema listovima koji se naziva *BroadcastStitch*. Na slici 5.4 prikazana je razlika između tih pristupa.

Prednost algoritma *ConvergecastStitch* je što u trenutku spajanja grozdova, čvorovi koji obavljaju spajanje imaju sve potrebne podatke, odnosno nema potrebe za dodatnim komunikacijskim opterećenjem. Komunikacijska složenost ovog algoritma je $\mathcal{M} = 2(n-1)$. Naime, duž svakog od $n-1$ bridova stabla potrebno je poslati dvije poruke, prva je zahtjev za početkom spajanja od korijena prema listovima, a druga je poruka kojom se grozdovi spajaju, od listova prema korijenu. Problem kod ovog pristupa je što, kako spajanje napreduje prema korijenu, (pod)grozdovi koji se spajaju su sve veći, te u konačnici korijen stabla spaja podgrozdove koji uključuju sve čvorove mreže.

S obzirom na upravo obrnuti smjer spajanja, *BroadcastStitch* metoda nema taj problem jer se njom uvijek spaja mali inicijalni grozd s velikim centralnim grozdom. Međutim, kako se centralni grozd širi konkurentno u svim smjerovima, postoji mogućnost da se u određenom trenutku ne koriste svi dostupni podaci. Kako bi se smanjila mogućnost ili u potpunosti izbjegla pojava takvih situacija, potrebna je dodatna komunikacija i osvježavanje podataka među čvorovima centralnog grozda. Komunikacijska složenost ovog algoritma iznosi $\mathcal{M} = \log n(n-1)$ u verziji u kojoj se u jednoj iteraciji spaja cijela jedna razina stabla ili $\mathcal{M} = n(n-1)$ u verziji u kojoj se u svakoj iteraciji spaja samo jedan, prema određenom kriteriju, odabrani grozd.

Prema kriterijima komunikacijske složenosti i veličine poruke, može se zaključiti kako je *BroadcastStitch* pogodniji za mreže s više čvorova jer, iako zahtjeva veći broj poruka, one su uvijek iste veličine bez obzira na broj čvorova u mreži, dok kod *ConvergecastStitch*, iako je ukupni broj poruka manji, njihova veličina je, kako se približavaju korijenu proporcionalna broju čvorova mreže.

Treći zadatak je odabir lokalnog redoslijeda, a njega je potrebno riješiti bez obzira na oblik stabla i na globalni poredak spajanja. Naime roditelj je uvijek zadužen za spajanje djece ili međusobno kod *ConvergecastStitch*-a ili s centralnim grozdom kod *BroadcastStitch*-a te on treba odlučiti o poretku tog spajanja. Kako odluku treba donijeti na osnovu podataka koji su mu lokalno dostupni, trenutno se koristi kriterij broja zajedničkih čvorova.

5.4 Heuristika za detekciju i izbjegavanje korištenja nedovoljno dobro estimiranih lokacija

Kao što se vidi iz gornjeg opisa algoritma, u nekoliko situacija čvor je prisiljen donijeti odluku na osnovu lokalno dostupnih podataka. Autori su u [49] analizirali podatke koji su čvoru dostupni, kao što su: procjena kvalitete prethodnih spajanja i korelacija između dvije formacije koje se spajaju, no nisu koristili procjenu kvalitete estimacije lokacija za formaciju koja se treba spojiti. S obzirom na odabrane algoritme tu odluku je potrebno

donijeti prilikom:

- određivanja položaja u inicijalnim grozdovima
- spajanja podgrozdova djece u stablu međusobno i s podgrozdovima roditelja te
- spajanja podgrozdova potomaka u stablu s podgrozdovima predaka odnosno unutarnjeg spajanja podgrozdova u korijenu.

Odluke koje se mogu donositi spadaju u dvije kategorije:

Filtar – izbjegavanje korištenja podgrozdova s nedovoljno dobro estimiranim lokacijama,

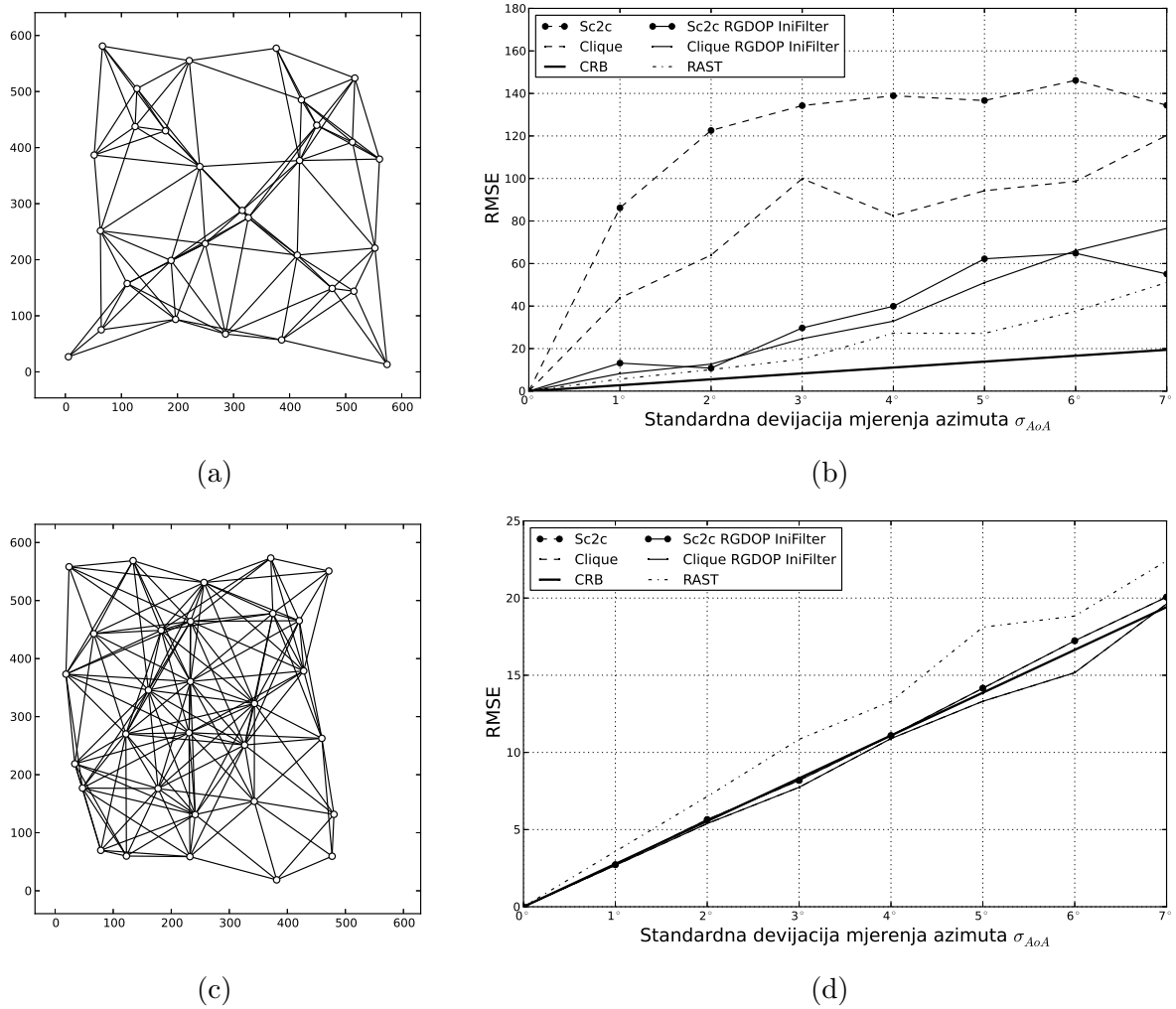
Redoslijed – prioritet pri korištenju imaju dijelovi mreže s kvalitetnije estimiranim lokacijama.

Osnovna ideja heuristike iz hipoteze je koristiti procjenu kvalitete estimacije položaja odnosno RGDOP, izvedenu u prethodnom poglavlju kako bi se donijela što bolja odluka. U ovom radu analizira se filter tip odluke prilikom određivanja položaja u inicijalnim grozdovima. Motivi za ovakvu definiciju heuristike su sljedeći: inicijalni grozdovi se zbog potencijalne relativno male povezanosti sastoje od malog broja čvorova te postoji mogućnost da je njihova formacija izrazito nepovoljna, te da dođe do multipliciranja pogreške mjerenja. Također, inicijalno određivanje položaja je početna i kritična faza algoritma čiji rezultati se koriste u narednim fazama. Zbog toga je izuzetno bitno izbjegavanje velikih pogrešaka koje u fazi spajanja mogu propagirati i uzrokovati nedovoljno dobru estimaciju cijelih dijelova mreže.

5.5 Rezultati simulacija

Kako bi se ustanovilo koliki je kvantitativni utjecaj odabrane heuristike obavljene su simulacije algoritma s i bez korištenja procjene, odnosno donošenja odluke o nekorištenju nedovoljno kvalitetnih inicijalnih podgrozdova. Prilikom postavljanja algoritma jedan od parametara koji se treba odrediti je i prag za vrijednosti RGDOP-a iznad koje se inicijalni podgrozd neće koristiti. Procijenjena pogreška dobivena iz RGDOP-a ne ovisi o standardnoj devijaciji mjerenja, niti o apsolutnoj veličini formacije, stoga je moguće odabrati generički prag koji bi se mogao koristiti u mrežama s vrlo različitim topologijama. Za određivanje konkretne vrijednosti praga koja se koristi u simulacijama korišteni su rezultati sa slika 4.9 i 4.11, te je za formacije s do 6 čvorova odabrana vrijednost 2. Za formacije s više od 6 čvorova vrijednosti RGDOP-a su, zbog većeg broja mjerenja, malene, pa su te formacije uvijek korištene u daljnjim fazama. Usporedba je obavljena između Cramér-Rao limita, centraliziranog algoritma RAST, te raspodijeljenih algoritama s podjelom podgrozdova korištenjem *Sc2c* i *Clique* metode bez korištenja i s korištenjem heuristike RGDOP inicijalnog filtra. Topologija ispitnih mreža i rezultati simulacija pri-

kazani su na slici 5.5.



Slika 5.5: Korijen srednje kvadratne pogreške (RMSE) dobiven koristeći jednadžbu (4.33) u simulacijama centraliziranog algoritma RAST i raspodijeljenog algoritma s različitim parametrima. Na slikama je naznačen i Cramér-Rao limit – CRB. Ispitne mreže su konveksne topologije, a prosječan broj susjeda po čvoru je 7 (a,b) odnosno 11 (c,d).

Iz dobivenih rezultata vidljivo je kako je primjena heuristike zasnovane na metodi procjene kvalitete estimacije RGDOP, kod mreže s manjim brojem susjeda po čvoru, omogućila znatno točniju estimaciju od algoritma koji ne koristi filtriranje inicijalnih podgrozdova. Može se zaključiti kako je uzrok velikim pogreškama, koje čini algoritam bez korištenja heuristike, u činjenici da koristi sve podgrozdove – bez obzira na moguću pojavu velikih pogrešaka koje spajanjem mogu propagirati. Utjecaj podjele inicijalnih grozdova je zanemariv na konačan rezultat, posebice u slučajevima kada se koristi heuristika.

Dodatno, može se uočiti kako centralizirana inačica algoritma omogućava najmanju pogrešku estimacije, tek nešto veću od Cramér-Rao limita. Centralizirani algoritam koji se koristio u simulacijama je algoritam RAST u svom izvornom obliku predstavljenom

u [47] odnosno u prilogu P1. Takvi rezultati su očekivani s obzirom da se centralizirani algoritam koristi svim raspoloživim mjerenjima u jednom složenom izračunu, a koji uključuje rastav po singularnim vrijednostima matrice dimenzija $(2 \cdot |\mathcal{E}| + 2) \times 2|\mathcal{V}|$. U konkretnom slučaju mreže s grafom koji sadrži 105 bridova i 30 vrhova dimenzije matrice iznose 212×60.

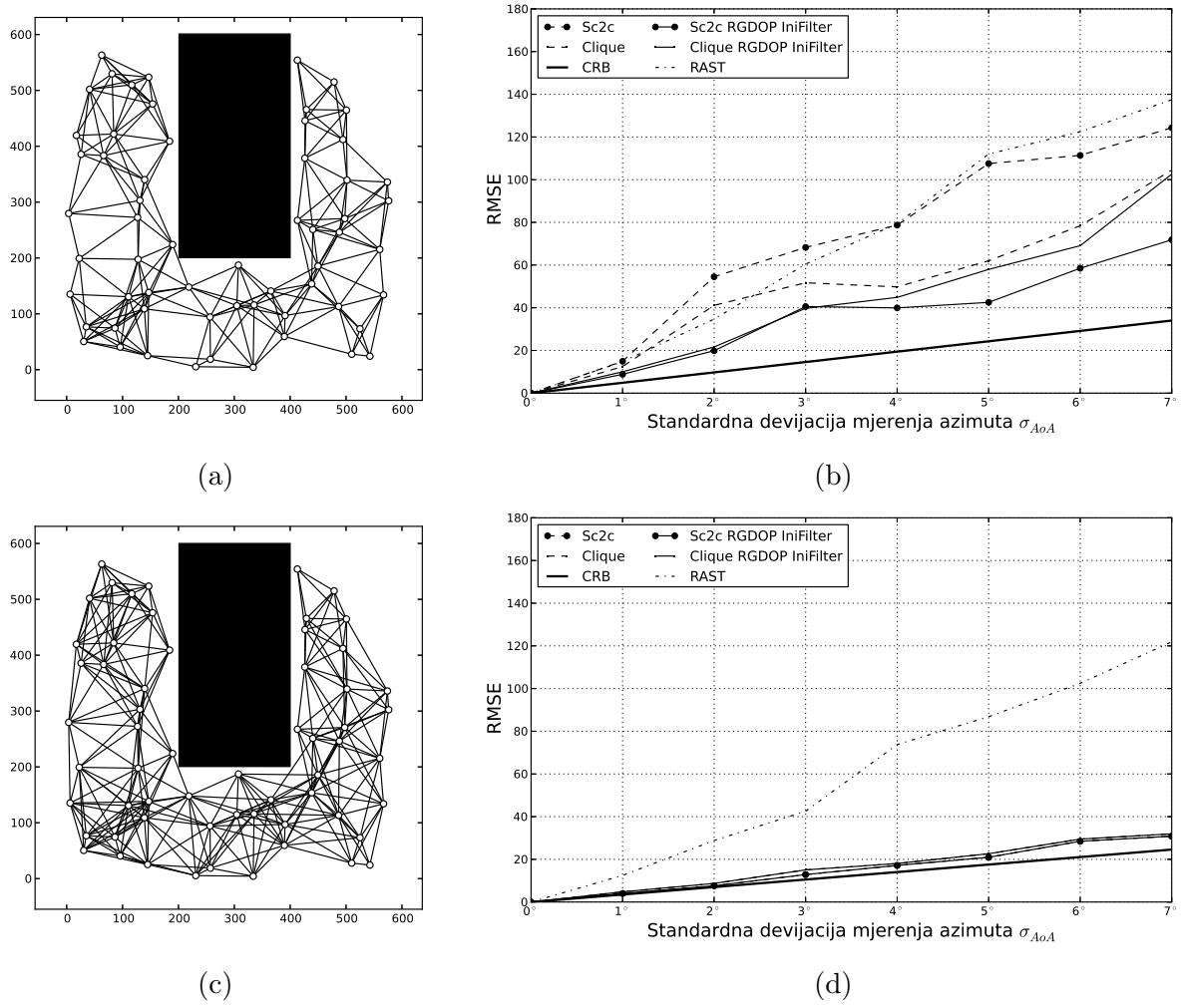
Naposlijetku, može se zaključiti kako raspodijeljeni algoritam uz korištenje heuristike može ostvariti rezultate usporedive s centraliziranom inačicom, algoritma, uz zadržane sve prednosti koje pruža takav način funkcioniranja. Za mreže s većim prosječnim brojem susjeda po čvoru, u simulacijama 11, raspodijeljeni algoritam nadmašuje centralizirani. Utjecaj heuristike se u tom slučaju gubi, što je i očekivano jer, kao što je spomenuto, RGDOP za grozdove s puno susjeda je ispod postavljenog praga.

5.5.1 Utjecaj nekonveksnih topologija

Analiza utjecaja nekonveksnih topologija na kvalitetu estimacije, obavljena je kroz simulacije algoritma za mreže postavljene u nekonveksnim prostorima. Na slici 5.6 prikazane su ispitne mreže s takvom topologijom s 8, odnosno 11 susjeda po čvoru te rezultati simulacija.

Za mrežu s manjim prosječnim brojem susjeda po čvoru utjecaj heuristike je najosjetniji. Razlog tomu kao i kod konveksnih topologija, leži u relativnom malom broju veza što može dovesti do većeg udjela formacija s većim iznosom RGDOP-a, pa filtriranje po tom kriteriju omogućava korištenje formacija koje imaju veću mogućnost za dobru estimaciju položaja. Kod mreža s većom povezanosti ovaj utjecaj se gubi. S druge strane još uvijek je izražen doprinos raspodijeljene inačice algoritma koja ima značajno manju pogrešku od centralizirane, vrlo blisku Cramér-Rao limitu.

Naposlijetku, treba naglasiti kako se svaki raspodijeljeni algoritam može učiniti centraliziranim, te bi rezultati u tom slučaju bili jednaki. S druge strane, i dalje bi raspodijeljena inačica algoritma zadržala sve ostale spomenute prednosti u odnosu na centraliziranu inačicu.



Slika 5.6: Korijen srednje kvadratne pogreške (RMSE) dobiven koristeći jednadžbu (4.33) u simulacijama centraliziranog algoritma RAST i raspodijeljenog algoritma s različitim parametrima. Na slikama je naznačen i Cramér-Rao limit – CRB. Ispitne mreže su nekonveksne topologije, a prosječan broj susjeda po čvoru je 8 (a,b) odnosno 11 (c,d).

Poglavlje 6

Eksperimentalna verifikacija

Uz ispitivanja u simulatoru, verifikacija algoritma provedena je kroz implementaciju na ispitnoj bežičnoj mreži osjetila. Ispitna mreža sastoji se od čvorova komercijalnog naziva JeeNode projektiranih oko mikrokontrolera Atmel ATmega328p i radio modula RFM12B. Izabrani čvorovi su vrlo ograničenih računalnih, a posebice memorijskih kapaciteta, stoga je bilo potrebno adaptirati pojedine dijelove algoritma kako bi se mogao izvršavati. Jedna od osnovnih pretpostavki algoritma za estimaciju položaja je mogućnost estimacije azimuta susjednih čvorova u mreži. U tu svrhu, u sklopu eksperimentalne verifikacije dodatno je isprojektirano, izrađeno i ispitano osjetilo za mjerenje azimuta koje se koristi određivanjem smjera dolaska svjetlosnog signala u infracrvenom dijelu spektra. Na kraju poglavlja dani su rezultati eksperimenata, s naglaskom na usporedbu s rezultatima dobivenim u simulatoru.

6.1 Odabrana platforma

Čvorovi JeeNode nastali su kao prilagodba Arduino platforme s kojom su održali kompatibilnost kroz korištenje istog razvojnog okruženja (engl. *Integrated Development Environment – IDE*) te programske knjižnice. Pojavom Arduina 2005. godine značajno je olakšan i financijski i vremenski ulazak u svijet mikrokontrolera što je dovelo do njegove velike popularnosti te velikog broja raznih proširenja i modifikacija za razne namjene.

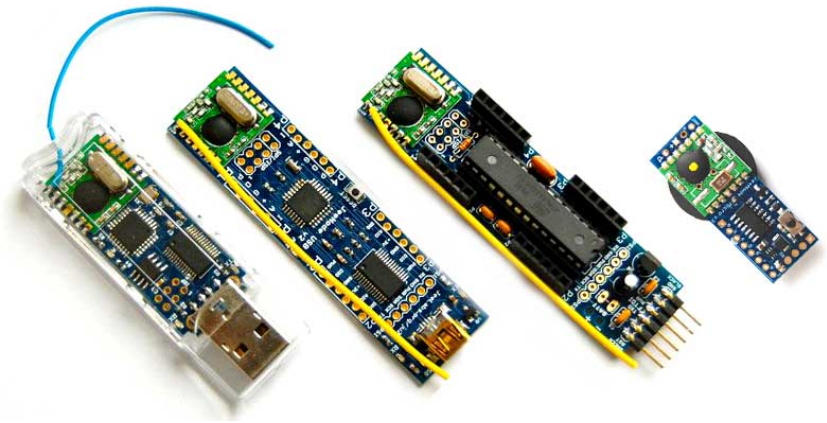
Osnovna motivacija za nastanak JeeNode platforme je profiliranje Arduina za upotrebu u bežičnim mrežama osjetila. Kako bi se podržala paradigma jeftinih bežično umreženih čvorova s niskom potrošnjom energije uvedene su sljedeće promjene:

- uključen je radiofrekvencijski modul kako bi se omogućila bežična komunikacija između mikrokontrolera
- smanjen je napon napajanja s 5 V na 3,3 V, prvenstveno kako bi se prilagodio nazivnom naponu za radio modul, a zatim i velikom broju osjetila koji funkcioniraju

na 3,3 V, te naposlijetku omogućilo napajanje litijskim baterijama s naponom od 3,5 do 4,2 V

- cjelokupni koncept je zasnovan na modelu niske potrošnje energije što se ponajviše očituje u programskim knjižnicama posebice u upravljačkom programu za modul za bežičnu komunikaciju
- za lakše spajanje s dodatnim modulima uvodi se koncept vrata (engl. *port*) sa 6 priključnica
- omogućeno je ulančavanje (engl. *daisy chaining*) dodatnih modula korištenjem I^2C protokola
- dimenzije platforme su minimizirane, no i dalje je omogućen izbor nekoliko različitih načina spajanja dodatnih modula.

Platforma dolazi u nekoliko formata prikazanih na slici 6.1 kao što su Jeelink koji je u eksperimentu korišten kao kontrolni čvor za osluškivanje i evidentiranje paketa za vrijeme izvršavanja algoritma ili JeeMicro koji je opremeljen slabijim modelom mikrokontrolera i s manje priključaka, ali puno manjih dimenzija.

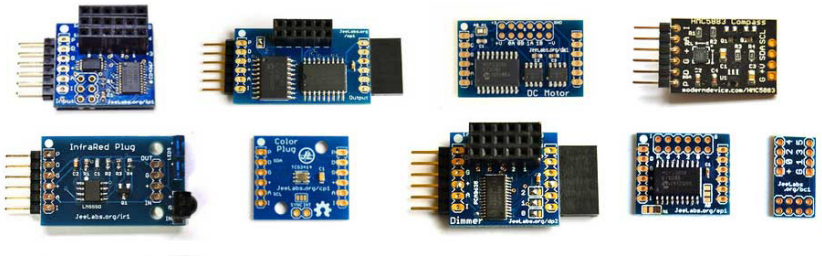


Slika 6.1: JeeNode platforma u različitim izvedbama. S lijeva na desno: JeeLink, JeeNodeSMD, JeeNode v6 i JeeMicro.

Ono što JeeNode platformu čini posebno zanimljivom za edukacijske, ali i istraživačke namjene je dobavljivost velikog broja različitih osjetila, aktuatora i ostale opreme koja se spaja na spomenute priključke. Oprema (slika 6.2) je cjenovno povoljna, sve sheme su javno dostupne, a svakom komadu opreme pridružen je odgovarajući razred u programskoj knjižnici te primjeri koji demonstriraju njihovo korištenje.

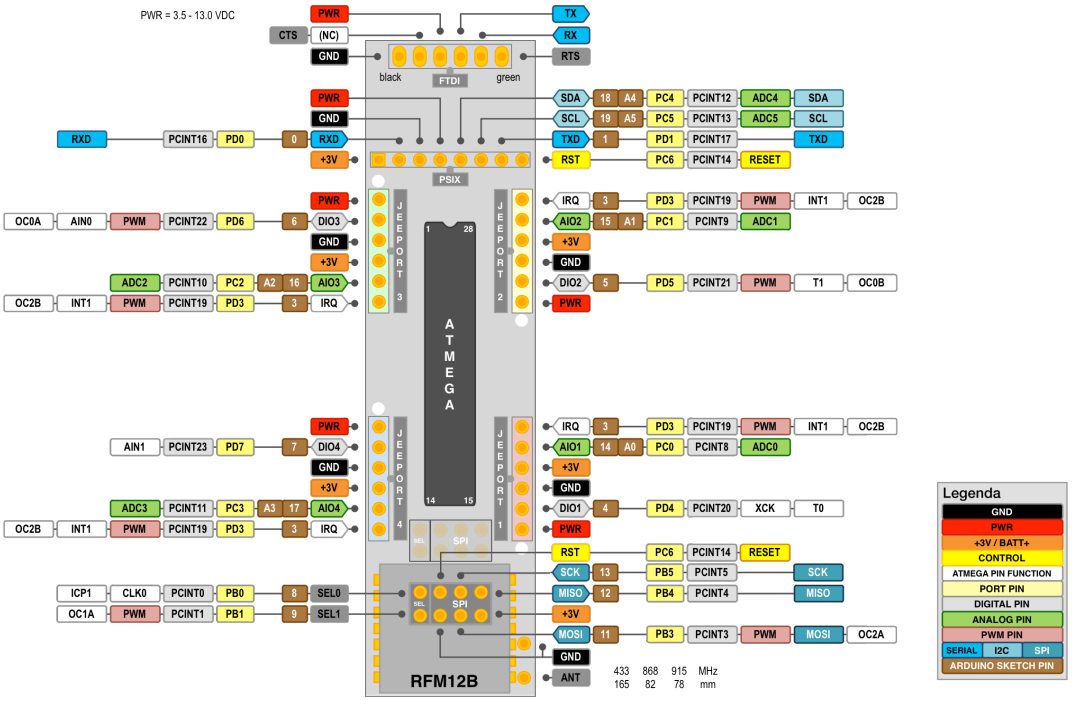
6.1.1 Mikrokontroler

Mikrokontroler koji čini osnovu čvorova mreže je Atmel ATmega328p. Kao i ostali mikrokontroleri iz porodice AVR zasnovan je na modificiranoj Harvard arhitekturi s 8 bitnim mikroprocesorom RISC (engl. *Reduced Instruction Set Computer*) arhitekture, s izabra-



Slika 6.2: Različita oprema kompatibilna s JeeNode platformom. Gornji red: Input Plug, Output Plug, DC Motor plug, Compass Board. Donji red: InfraRed Plug, Color Plug, Dimmer, Expander Plug

nom radnom frekvencijom od 16 MHz. Konkretni mikrokontroler sadrži 32 kB memorije za pohranu programa, 1024 byte EEPROM-a namijenjenog uglavnom za pohranu konfiguracijskih parametara, te 2048 byte-a radne memorije. Mikrokontroler je opremljen nizom funkcija kao što su brojila, kanali s mogućnošću generiranja impulsno širinskih signala (engl. *Pulse Width Modulation – PWM*), SPI, I^{2}C i USART sučelja.



Slika 6.3: Oznake priključaka na JeeNode platformi prema različitim formatima kojima su referencirani u dokumentaciji i programskom kôdu.

Za implementaciju algoritma posebno je važan 10-bitni analogno digitalni pretvornik koji se koristi kod mjerenja azimuta, te mogućnost detekcije i obrade zahtjeva za prekid (engl. *interrupt detection*) koja se koristi za asinkrono slanje i primanje poruka putem radio veze. Mikrokontroler podržava 6 različitih načina rada s niskom potrošnjom energije, što je, iako u algoritmu nije iskorišteno, vrlo bitno s aspekta funkcioniranja bežične mreže osjetila kod koje čvorovi, prema definiciji, imaju ograničene energetske resurse.

Mikrokontroler sadrži i 23 ulazno-izlazne linije od kojih je većini dodijeljena uloga te su vezani na priključke JeeNode platforme kao što je prikazano na slici 6.3.

S obzirom na arhitekturu mikrokontrolera, dva su velika izazova s kojima se susrelo tijekom implementacije. To su vrlo ograničena radna memorija i nepostojanje jedinice za operacije s pomičnim zarezom (engl. *Floating Point Unit – FPU*). Oba izazova riješena su korištenjem programske knjižnice AVR Libc [69] u kojoj su između ostalog implementirane funkcije za dinamičku alokaciju memorije te za operacije s pomičnim zarezom posebno prilagođene AVR seriji mikrokontrolera. S obzirom na zadane parametre problema, memorijski dio je riješen optimalno, no operacije s pomičnim zarezom bi se mogle dodatno optimirati korištenjem prikladnije aritmetike s nepomičnim zarezom.

6.1.2 Radio modul

Za bežičnu komunikaciju između JeeNode čvorova koristi se radio modul RFM12B. Modul omogućava prijam i odašiljanje radio signala u frekvencijskom području za industrijske, znanstvene i medicinske primjene (engl. *Industrial, Scientific and Medical Applications – ISM*), odnosno 433 i 915 MHz, te u području namijenjenom za uređaje kratkog dometa (engl. *Short Range Devices – SRD*) bez potrebe za ishodovanjem dozvole, odnosno 868 MHz koji je korišten i u samom eksperimentu. Za prijenos podataka koristi se diskretna modulacija frekvencije (engl. *Frequency Shift Keying – FSK*) s mogućim podešavanjem razmaka među frekvencijama od 15 do 240 kHz te širinom pojasa prijamnika od 67 do 400 kHz što omogućava brzine prijenosa do 115,2 kb/s. U samom eksperimentu korištene postavke omogućavaju brzinu prijenosa od 49,261 kb/s što predstavlja kompromis između brzine i pouzdanosti.

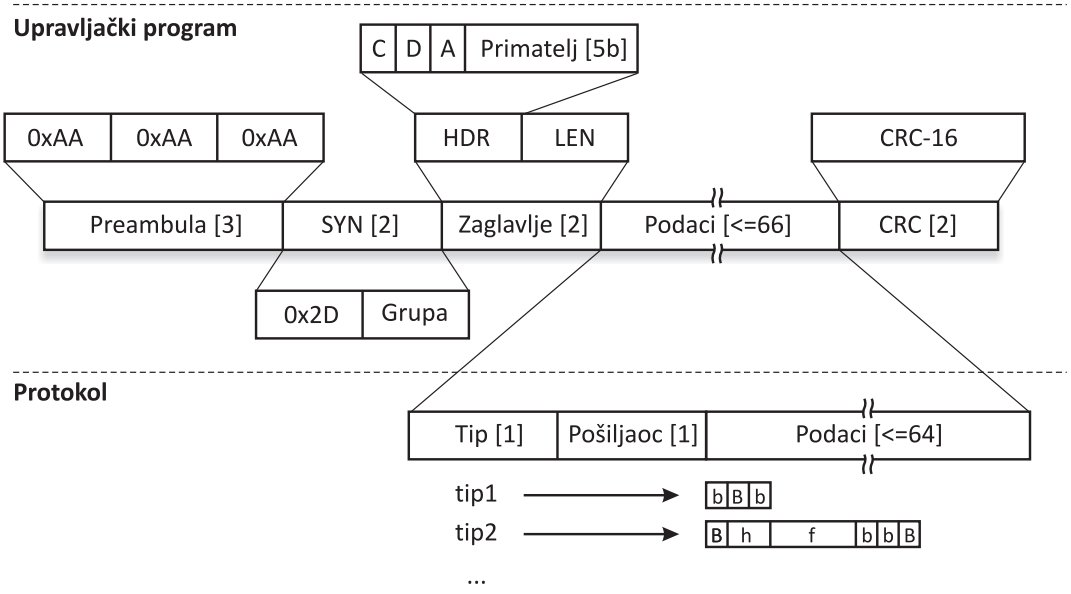
Od ostalih karakteristika može se izdvojiti indikator jakosti primljenog signala (engl. *Received Signal Strength Indicator – RSSI*) koji se, kao što je spomenuto u dijelu 2.1.1, može iskoristi za estimiranje udaljenosti između prijamnika i odašiljača. Modul podržava dva oblika indikacije jakosti signala: digitalni, u kojem se može postaviti prag te se zatim u statusnom registru očitava je li jakost signala ispod ili iznad postavljenog praga, te drugi, za navedenu namjenu korisniji, analogni. U analognom načinu se na jednom od kondenzatora na pločici (priključak 15) može očitati napon koj je proporcionalan jakosti signala u rasponu od -100 dBm do -65 dBm s deklariranom točnosti od ±6 dBm.

Domet modula je oko 50-70 m na otvorenom prostoru. U zatvorenim prostorima je manji, ali još uvijek dovoljno velik da obuhvati cijelu testnu mrežu.

6.1.3 Programske knjižnice i upravljački programi

Uz prethodno spomenuti AVR Libc, za implementaciju algoritma koriste se razredi i funkcije definirane u Arduino knjižnici te u JeeLib knjižnici koja je posebno namijenjena za JeeNode platformu i dodatne module. Od korištenih funkcija iz Arduino knjižnice može se izdvojiti razred Serial za serijsku komunikaciju s računalom koji je u početnim fazama implementacije pružao sučelje za kontrolu i ispitivanje čvorova. Serijska veza je kasnije u potpunosti izbačena iz upotrebe te zamijenjena prikladnijim sučeljem koje se koristi radio kanalom.

Programska podrška za radiokomunikaciju na JeeNode platformi implementirana je kroz upravljački program za radio modul RFM12B, a koji predstavlja okosnicu knjižnice JeeLib. Iako se za istu namjenu mogu koristiti i alternativne, funkcijama bogatije knjižnice kao što je LowPowerLab, JeeLib je referentna knjižnica i pruža dovoljno funkcija za definiranje namjenskih protokola, što je i iskorišteno.



Slika 6.4: Struktura paketa prema definiciji u upravljačkom programu unutar JeeLib knjižnice, te struktura podatkovnog dijela definiranog unutar protokola odnosno samog algoritma.

Upravljački program pretpostavlja korištenje posebno definiranog formata paketa prikazanog na gornjem dijelu slike 6.4. Paket započinje konstantnim ponavljajućim uzorkom bitova 101010... duljine 3 byte-a nakon čega slijede 2 byte-a za sinkronizacijsku sekvencu (SYN). Unutar te sekvence drugi byte označava adresu grupe, a kako sinkronizacijsku sekvencu obrađuje sam radio modul, mikrokontroler je oslobođen tog zadatka. Daljnje adresiranje unutar pojedine grupe izvršava se u mikrokontroleru koji odbacuje sve poruke koje nisu namjenjene za dani čvor što je definirano u zaglavlju paketa koje se sastoji od 2 byte-a. U prvom byte-u je s 5 bitova definiran identifikacijski broj čvora

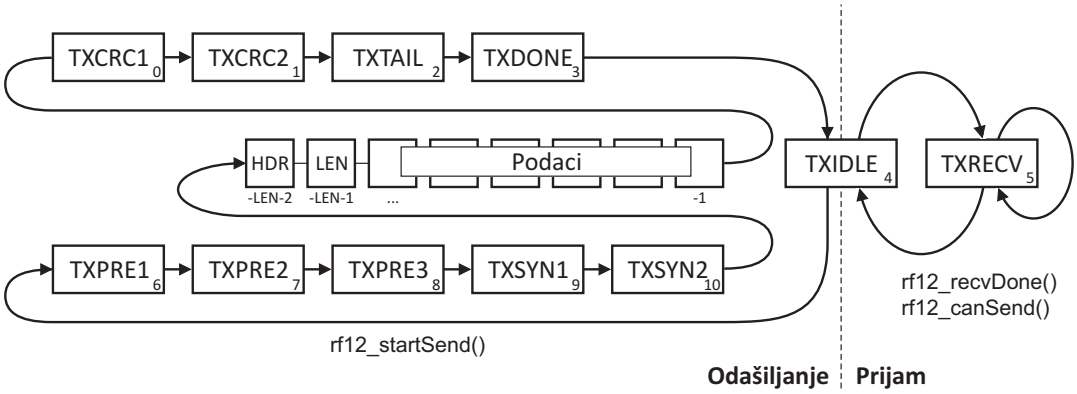
primaoca, odnosno njegova adresa unutar grupe. Ostala tri bita redom s lijeva na desno su: CTL - vrijednost 0 označava da je to normalan paket, a 1 da je to poseban paket potvrde primitka (engl. *acknowledgment*), DST bit za vrijednost 0 određuje da je to paket namijenjen svim čvorovima u grupi (engl. *broadcast packet*) (tada se ID čvora primaoca, zamijenjuje s ID brojem pošiljaoca), a vrijednost 1, da je namijenjen određenom čvoru, te bit ACK kojim se definira da li se za paket traži li se slanje potvrde primitka od strane čvora primaoca. S obzirom na ovako definiranu strukturu paketa gdje je za grupu rezrviran 1 byte, a za ID čvora 5 bitova, ukupno je moguće adresirati 256 različitih grupa s ukupno 31 čvorom u svakoj grupi (adresa čvora 0 se koristi za posebne namjene). Drugi byte zaglavlja označava duljinu podatkovnog dijela koja može iznosti od 0 do 66 byte-a, te na samom kraju nalaze se 2 byte-a kojima se paket štiti od pogrešaka u prijenosu koristeći CRC-16 (engl. *Cyclic Redundancy Check*) algoritam.

Osim funkcije adresiranja među grupama, i unutar grupe, provjere ispravnosti korištenjem CRC-a, te jednostavnog paketa potvrde (*ack*) i opcionalnog kriptiranja, upravljački program ne omogućava funkcije viših komunikacijskih slojeva kao što su primjerice održavanje veze s kraja na kraj odnosno usmjeravanje prometa i prosljeđivanje paketa čvorovima koji nisu u dometu. Takve funkcije, ukoliko su potrebne, treba posebno implementirati unutar samog programa. Ovakva situacija u konkretnoj implementaciji ne predstavlja problem jer jedina konceptualna pretpostavka raspodijeljenog algoritma je komunikacija između čvorova u dometu, a za to je dovoljno korištenje paketa potvrde u sprezi s pseudoslučajnim odgađanjem slanja u slučaju izostanka primitka potvrde.

Ono što ovakvom definicijom paketa nedostaje je identifikacija pošiljaoca u slučaju kada je paket namijenjen određenom čvoru, te definicija strukture podataka koja se šalje u samom paketu. Oba nedostatka su riješena dodavanjem posebnog zaglavlja u podatkovnom dijelu kao što je prikazano na donjem dijelu slike 6.4. Dodavanjem byte-a kojim se određuje struktura paketa moguće je slanje 256 različitih tipova poruka, a obaveznim slanjem ID broja pošiljaoca između ostalog se omogućava slanje paketa potvrde s definiranim primaocem. Obje promjene implementirane su unutar posebnog razreda `DisAlgProtocol` koji uz spomenuto zaglavlje definira posebne metode za slanje paketa s pseudoslučajnim odmakom kao što je detaljnije opisano u dijelu 6.3.

Osnovna karakteristika upravljačkog programa za radio modul je njegovo funkcioniranje u formi automata konačnih stanja čije akcije se izvršavaju u prekidnim potprogramima. Na taj način omogućava se izvršavanje glavnog programa paralelno s primanjem odnosno slanjem poruka. Glavni program u svakom trenutku može saznati u kojem je stanju upravljački program za radio modul te, u ovisnosti o stanju, izvršiti željenu akciju.

Na slici 6.5 prikazana su stanja u kojima se može nalaziti upravljački program te njihov kronološki slijed i akcije koje se obavljaju. Treba primijetiti nekoliko bitnih ka-



Slika 6.5: Dijagram stanja upravljačkog programa za radio modul. Program mijenja stanje prema prikazanom redoslijedu ili unutar prekidnog potprograma ili pozivom na neku od funkcija opisanih u nastavku. Tijekom osluškivanja, primanja, te nakon uspješnog primitka poruke program se nalazi u stanju TXRECV iz kojega izlazi isključivo pozivom funkcije `rf12_recvDone()` od strane glavnog programa. Ako je u tom trenutku primitak paketa već obavljen, što se identificira s uspješno primljenim i u međuspremniku (engl. *buffer*) zapisanim podacima duljine LEN + 5 tada program prelazi u stanje TXIDLE, a radio modul se gasi. Stanje TXIDLE je ključno jer u ovisnosti o tome koja se funkcija pozove sljedeća, ulazi se ili u sekvencu stanja za odašiljanje podataka na lijevoj strani, a odgovara strukturi paketa koji se u tom trenutku treba nalaziti u međuspremniku ili se prelazi natrag u stanje TXRECV za prijam. Ukoliko se treba poslati paket, a program je u stanju TXRECV, tada je potrebno pozvati funkciju `rf12_canSend()` koja provjerava prima li se trenutno poruka odnosno je li međuspremnik prazan i pokazuje li registar za digitalnu indikaciju jakosti signala RSSI kako je trenutna jakost signala ispod praga. Ako se poruka trenutno ne prima, tada se prelazi u stanje TXIDLE i daljnjim pozivom funkcije `rf12_sendStart()` u sekvencu za slanje. Ako se trenutno prima paket, potrebno je iterativno pozivati `rf12_canSend()` i `rf12_recvDone()` sve dok se paket u potpunosti ne primi te se stanje promijeni u TXIDLE. Paket koji je primljen na taj način može i ne mora biti obrađen, odluka ostaje na programu.

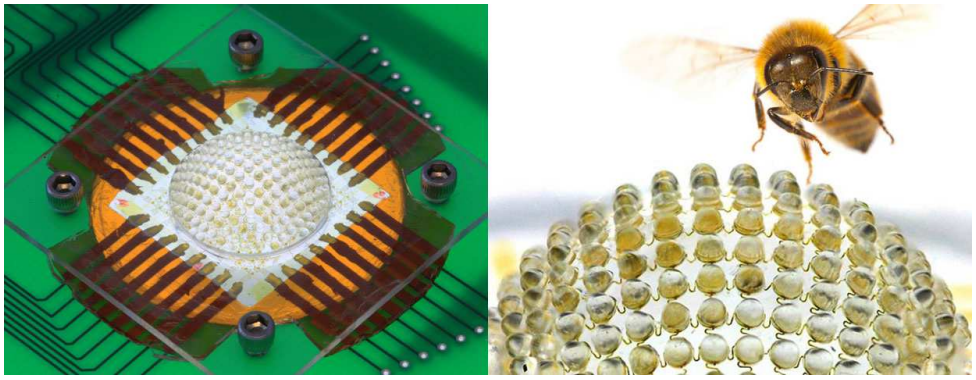
rakteristika radio modula i upravljačkog programa:

- u određenom trenutku može se isključivo ili primati ili slati pakete, ali ne obje akcije istovremeno
- posljedično za primanje i slanje poruka koristi se isti međuspremnik
- sadržaj paketa u međuspremniku se ne mijenja samo dok je program u stanju TXIDLE, osnosno potrebno ga je kopirati iz međuspremnika čim prije kako bi prijamnik mogao nastaviti primati odnosno slati pakete.

6.2 Osjetilo za mjerenje azimuta

Kako bi se mogla obaviti implementacija i verfikacija algoritma na ispitnoj mreži osjetila, jedan od ključnih dijelova opreme koji je potrebno osigurati je osjetilo za mjerenje azimuta. Gotovo, komercijalno dostupno rješenje ne postoji, a kao što je već spomenuto u dijelu 2.1.2 nedostaci metoda obrađenih u literaturi, a koji se zasnivaju na ultrazvučnim

i radiofrekvencijskim signalima, su veličina i/ili potreba za pokretnim dijelovima. Kao inspiracija za izradu novog tipa osjetila koji bi za određivanje azimuta koristio svjetlosne signale poslužilo je istraživanje [70] u kojem je predstavljen novi tip digitalne kamere. Kamera je veličine 1cm u promjeru i sadrži ukupno 180 mikroleća orijentiranih u različitim smjerovima kao što je prikazano na slici 6.6 preuzetoj iz [10]. Ovakav bioinspirirani model kamere s elementima na hemisferi pruža dvije velike prednosti u odnosu na klasični planarni model: (1) usprkos malim dimenzijama pokriva se vrlo širok kut, trenutno oko 160° te (2) postiže se vrlo velika dubinska oštrina koja omogućava zadržavanje objekta u fokusu neovisno o udaljenosti od kamere.



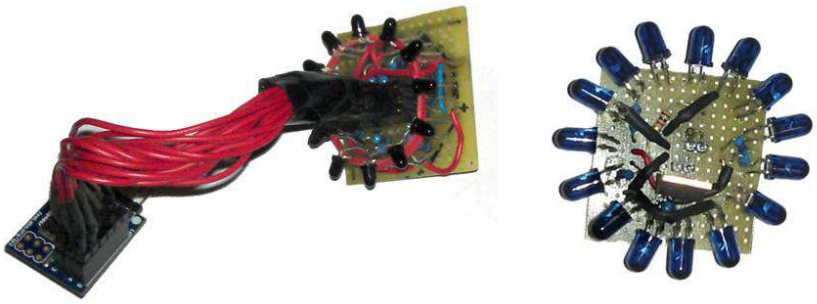
Slika 6.6: Bioinspirirana digitalna kamera s fotoelementima i mikrolećama na hemisferi [10].

6.2.1 Izvedba

Osnovna ideja osjetila za mjerenje azimuta je omogućiti detekciju svjetlosnog signala iz svih smjerova te na osnovu jakosti odrediti smjer dolaska signala. Kako bi se omogućilo dvostrano mjerenje, svako osjetilo sadrži i odašiljačku stranu koja služi kao kratkotrajni izvor infracrvenog zračenja čiji azimut se određuje na drugom čvoru koristeći prijamnu stranu osjetila. Prototip osjetila prikazan na slici 6.7 izrađen je koristeći InputPlug modul za JeeNode platformu, a koja omogućava multipleksiranje 16 analognih signala. Prijamni i odašiljački dio sastoji se od fotodioda odnosno fototranzistora koji funkcioniraju u infracrvenom dijelu spektra.

Ispitivanja prototipa ukazala su na potencijal ove metode. Već s prvim rudimentarnim algoritmom za estimaciju azimuta iz očitanih vrijednosti napona s otpornika serijski spojenih s fototranzistorima dobivene pogreške bile su ispod 10°.

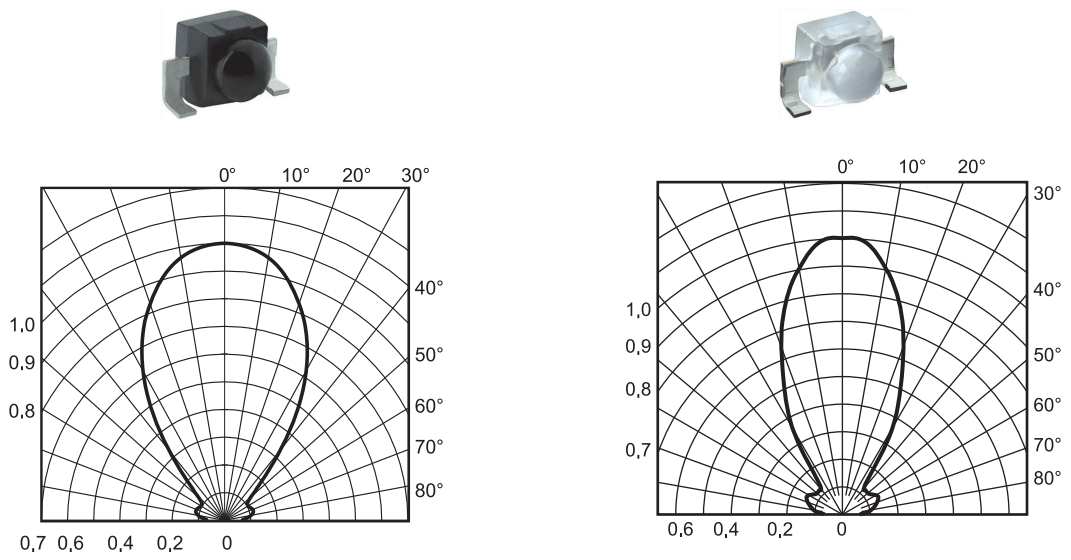
Ova inačica poslužila je i za identifikaciju potencijalnih unaprijeđenja za iduću verziju. Željena unaprijeđenja uključuju poboljšani proces izrade pločice i ugradnje elemenata kako bi se postigla preciznija geometrija, te smanjenje dimenzija i integraciju postojeća tri sklopa (InputPlug, odašiljački dio i prijamni dio) u jedan jedinstveni sklop. Također rezultati ispitivanja prvog prototipa pokazali su kako veliki dio ukupne pogreške



Slika 6.7: Prototip osjetila za mjerenje azimuta.

čine sustavne greške te se moglo zaključiti kako bi jedna od korisnih intervencija bila omogućiti umjeravanje i upisivanje dobivenih umjernih parametara na osjetilo. Na osjetilu se već nalazi mikrokontroler Atmel ATTiny45 koji služi za selekciju kanala na multiplekseru, čija memorija se može iskoristiti za upisivanje umjernih parametara i posljedično smanjenje pogreške estimacije azimuta.

S obzirom na zadane specifikacije osjetilo je izrađeno u tehnologiji površinske ugradnje (engl. *Surface Mount Technology – SMT*) na dvostranoj pločici. Na donjoj, prijamnoj strani pločice nalazi se multiplekser, mikrokontroler te fototranzistori, a na gornjoj odašiljačkoj, fotodiode.



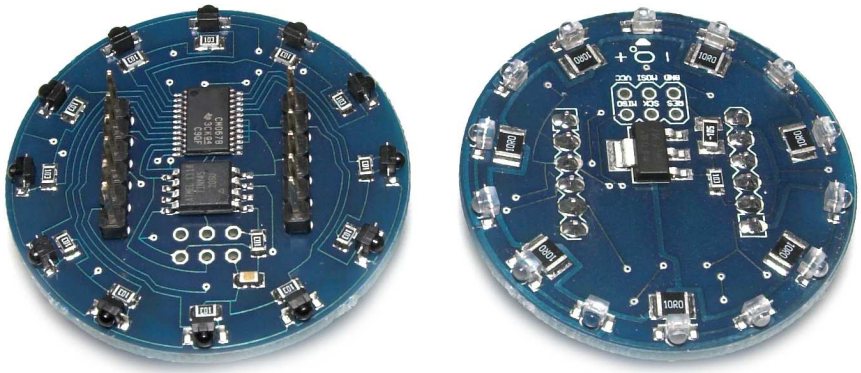
(a) Fototranzistor (b) Fotodioda

Slika 6.8: Izabrani elementi i njihovi dijagrami osjetljivosti/zračenja preuzeti iz dokumentacije [71, 72].

Uvjet izbora fotoelemenata je da budu malih dimenzija, namijenjeni za površinsku ugradnju te da se mogu ugraditi pod kutem od 90 stupnjeva. Za fototranzistore i diode odabrani su Vishay VEMT2023SLX01 i VSMB2943SLX01 koji su usklađeni po valnoj duljini. Fotodiode su velike snage, a dijagram zračenja im je takav da ih je dovoljno

14 pravilno raspoređenih po obodu osjetila kako bi se omgućilo ravnomjerno zračenje u horizontalnoj ravnini u svim smjerovima. Kako prikazuje slika 6.8, dijagram osjetljivosti fototranzistora je takav da je -3 dB širina dijagrama jednaka $\pm 35°$ što, ukoliko se koristi 12 pravilno raspoređenih fototranzistora, omogućava detekciju signala s barem 3 fototranzistora bez obzira na kut dolaska signala.

Osjetilo je kružnog oblika s foto elementima na obodu, a spaja se konektorima na dva susjedna vrata JeeNode čvora. Shema i raspored elemenata na pločici dani su u prilogu P3, a konačni izgled osjetila prikazan je na slici 6.9.

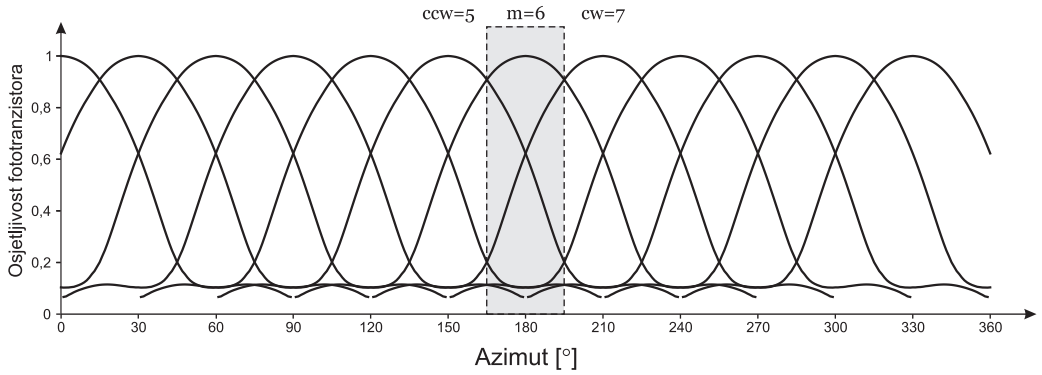


Slika 6.9: Konačna inačica osjetila za mjerenje azimuta, lijevo donja prijamna strana i desno gornja odašiljačka.

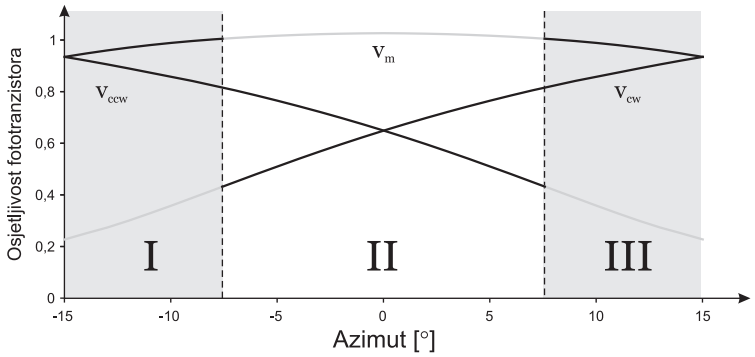
6.2.2 Algoritam za estimaciju

Analizom dijagrama zračenja prostorno razmaknutih i različito orijentiranih fototranzistora, kao što je prikazano na slici 6.10a, dolazi se do zaključka kako bi algoritam za estimaciju u prvoj fazi trebao napraviti selekciju tri fototranzistora s najvećim upadnim signalima odnosno izmjerenim naponima na odgovarajućim otpornicima. Naime, prema dijagramu, vrijednosti dobivene na ostalim fototranzistorima su preniske kako bi bile relevantne za estimaciju. Drugi zaključak je kako se apsolutne vrijednosti očitanja jakosti upadnog signala ne mogu iskoristiti, s obzirom da su ovisne o nepoznatoj udaljenosti između osjetilâ, te o ambijentalnom zračenju u infracrvenom dijelu spektra.

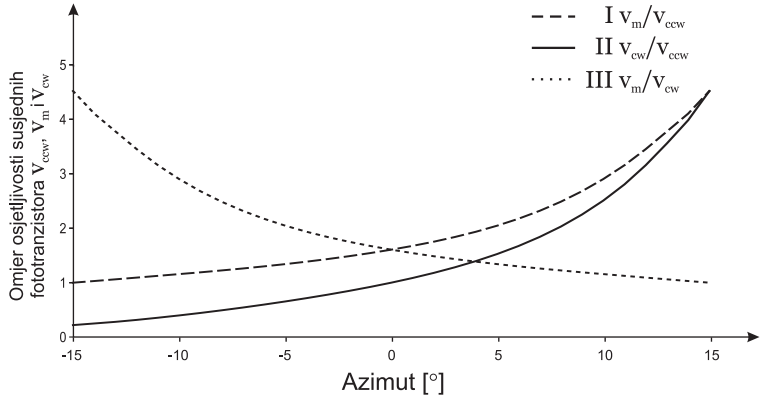
Uzimajući u obzir prethodno navedene zaključke, veličine koje se mogu iskoristiti za estimaciju azimuta su: omjeri vrijednosti dobivenih na tri fototranzistora s najvećim iznosom odnosno: v_m/v_{cw}, v_m/v_{ccw} te v_{cw}/v_{ccw}, ako su sa v_m, v_{cw} i v_{ccw} redom označeni iznosi mjerenja na fototranzistoru s maksimalnim očitanjem, susjednom fototranzistoru koji se nalazi u smjeru kazaljke na satu, te onom koji se nalazi u smjeru suprotnom od kretanja kazaljke na satu.



(a)



(b)



(c)

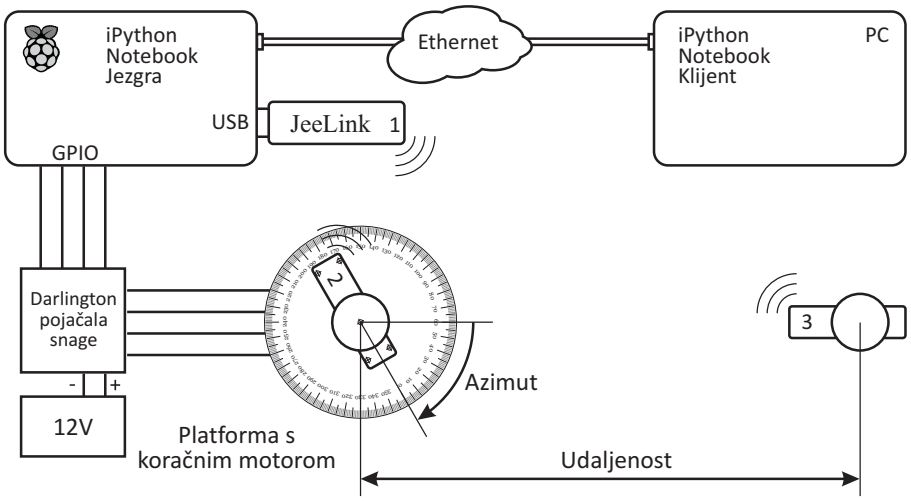
Slika 6.10: Krivulje koje koristi algoritam za estimaciju azimuta dobivene koristeći dijagram osjetljivosti fototranzistora iz dokumentacije.

Nakon što se definiraju tranzistori m, cw i ccw, daljnja estimacija azimuta obavlja se u rasponu od $-15°$ do $15°$, relativno na orijentaciju tranzistora m s maksimalnom izmjerenom vrijednosti. Koji od tri navedena omjera je najpogodnije koristiti ovisi u kojem segmentu navedenog raspona azimuta se estimira. Naime, prema krivulji na slici 6.10b, u većem dijelu raspona azimuta vrijednost v_m se mijenja tek neznatno dok se karakteristike za tranzistore cw i ccw nalaze u svom najstrmijem dijelu te omjer vrijednosti dobivenih na tim tranzistorima predstavlja idealnog kandidata za vrijednost iz koje se može estimi-

rati azimut. Segment u kojem će se koristiti omjer v_{cw}/v_{ccw} označen je sa II. Problemi s korištenjem tog omjera nastaju na rubovima raspona stoga što vrijednost v_{cw} na donjem segmentu raspona, onom označenom sa I, odnosno v_{ccw} na gornjem segmentu raspona, označeno, sa III, su preniske te šum kvantizacije utječe na točnost dobivenih vrijednosti. U tom slučaju pogodno je koristiti ostala dva omjera, stoga se u drugom koraku estimacije iz omjera v_{cw}/v_{ccw} određuje kojem segmentu pripada estimacija: I, II ili III, te ako pripada segmentu I ili III koriste se odgovarajući omjeri. Širine segmenata I i III su parametri algoritma za estimaciju, a u daljnjem ispitivanju za oba segmenta koristi se širina od 8°. Naime upravo taj iznos širine segmenata, kako se pokazalo na podacima dobivenim prilikom umjeravanja osjetila, omogućava najveću točnost estimacije.

Ispitivanje i umjeravanje

U dosadašnjoj analizi korišteni su isključivo podaci iz dokumentacije, no za realnu evaluaciju potrebno je obaviti ispitivanja na samom osjetilu. Kako bi se olakšalo i ubrzalo mjerenja, te posljedično i umjeravanje osjetila proces je potpuno automatiziran. Shematski prikaz ispitne platforme dan je na slici 6.11.



Slika 6.11: Shematski prikaz platforme za umjeravanje osjetila.

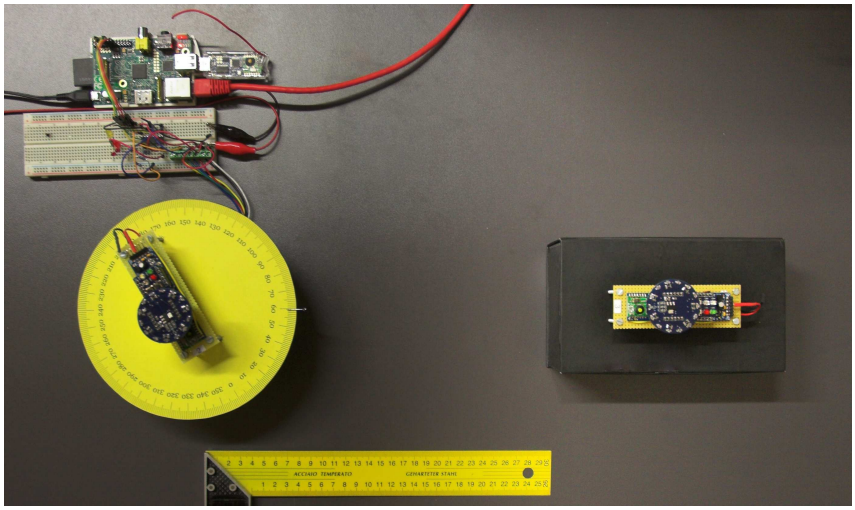
Središnji dio platforme čini RaspberryPi računalo na kojem je pokrenut iPython Notebook jezgra (engl. *kernel*). Unutar jezgre su instancirani razredi za serijsku komunikaciju preko koje se upravlja priključenim JeeLink čvorom, te razred za kontrolu GPIO (engl. *General Purpose Input Output*) priključaka spojenih preko pojačala snage na koračni motor. Ispitna platforma je upravljiva s osobnog računala koristeći iPython Notebook klijent, odnosno u konkretnom slučaju internet preglednik. Mrežni dio platforme sastoji se od tri čvora:

- JeeLink čvor, na shemi označen kao čvor 1, služi za slanje komandi čvoru 2 i 3 te za primanje mjernih podataka s čvora 2.
- Čvor 2 je postavljen na rotirajuću platformu i na njega je priključeno osjetilo koje se umjerava.
- Čvor 3 ima također priključeno osjetilo na kojem se koristi gornja, odašiljačka strana odnosno služi kao kontrolirani izvor infracrvenog zračenja na poznatoj udaljenosti i s poznatim azimutom u odnosu na čvor 2.

Mjerni protokol se sastoji od sljedećih koraka:

1. Čvor 1 šalje čvoru 2 poruku za mjerenjem ambijentalnog zračenja u infracrvenom frekvencijskom području i postavljanje praga.
2. Nakon što čvor 2 obavi inicijalno mjerenje, čvor 1 šalje čvoru 3 poruku da uključi diode, te odmah nakon toga čvoru 2 da započne mjerenje azimuta.
3. Nakon mjerenja čvor 2 šalje poruku s izmjerenim vrijednostima i estimiranim azimutom koju prima čvor 1.

Nakon obavljenog mjerenja koračni motor se zakreće za jedan polukorak odnosno, u konkretnom slučaju za 0,9° u smjeru kazaljke na satu, te se pokreće novo mjerenje. Proces se ponavlja za svih 360° stupnjeva odnosno 400 koraka. Dobiveni rezultati sastoje se od udaljenosti koja je ista za cijelo mjerenje, pravog azimuta dobivenog iz poznatog položaja koračnog motora odnosno platforme, 12×400 mjerenja s fototranzistora te estimirane vrijednosti azimuta.

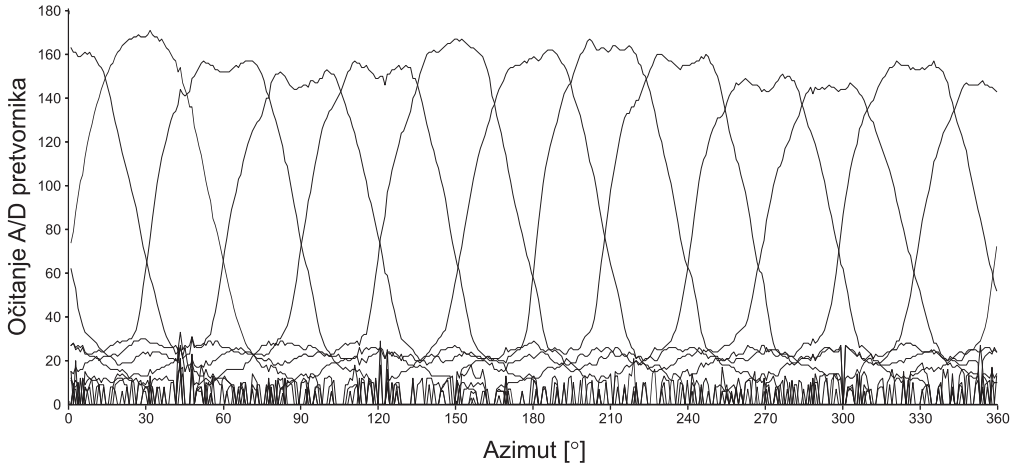


Slika 6.12: Ispitna platforma

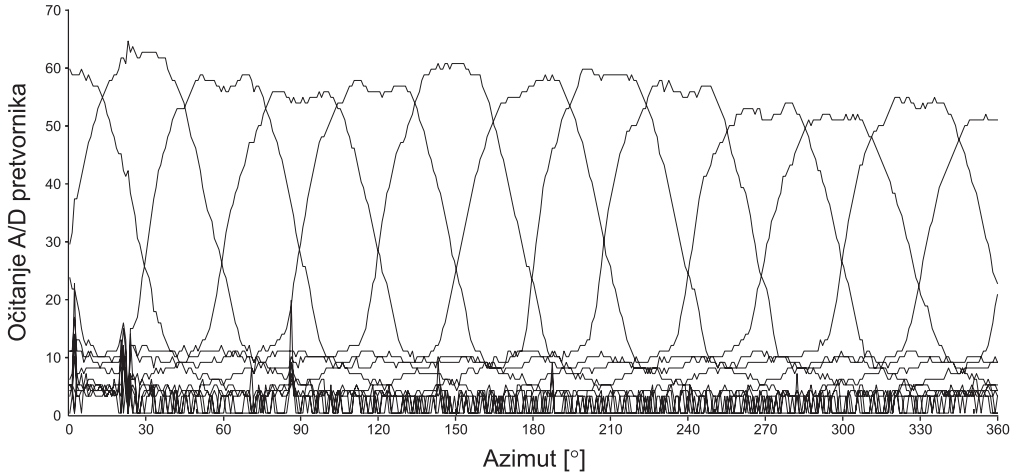
Rezultati ispitivanja

Na slici 6.13 prikazani su tipični rezultati ispitivanja osjetila za dvije različite udaljenosti. Na x osi je pravi azimut izvora zračenja, a na y osi očitana vrijednost napona na otpornicima serijski spojenih s pojedinim fototranzistorom. Izmjereni napon, osim o udaljenosti,

ovisi i o ambijentalnom osvjetljenju u infracrvenom dijelu spektra te o naponu napajanja, npr. napon litij ion baterija koje se koriste u eksperimentu imaju raspon od 3,3 V do 4,2 V. Bitno je primjetiti kako se na različitim udaljenostima dobivaju slične karakteristike, samo je apsolutni iznos drukčiji te je na većim udaljenostima posljedično veći utjecaj šuma kvantizacije. Iz rezultata se može uočiti kako se karakteristike pojedinih tranzistora u maksimumu prilično razlikuju od krivulja u dokumentaciji. Nakon provedene analize pretpostavka je kako su te, za svaki tranzistor jedinstvene karakteristike, uzrokovane nesavršenostima u procesu izrade kućišta odnosno leće.



(a) $d = 50$ cm



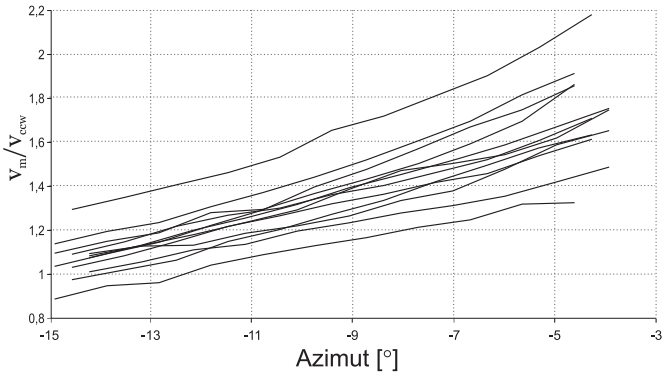
(b) $d = 80$ cm

Slika 6.13: Mjerenja pada napona na otpornicima serijski spojenih s fototranzistorima za dvije različite udaljenosti. Vrijednost napona prikazana je kao očitanje 10-bitnog A/D pretvornika.

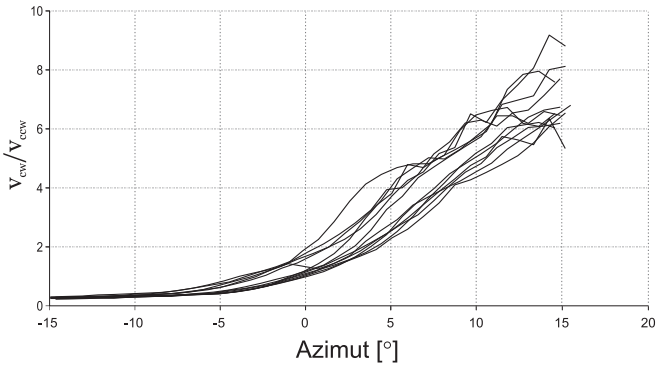
Spomenute karakteristike, koje se u pojedinim slučajevima poprilično razlikuju od dokumentiranih, kao i nesavršenosti u ugradnji odnosno orijentaciji pojedinog fototranzistora mogu uzrokovati pogreške u estimaciji. Promatrajući izmjerene karakteristike,

dobra strana predloženog algoritma je što se u segmentu II u kojem karakteristika ima najveći odmak od idealne njene vrijednosti ne koriste za estimaciju.

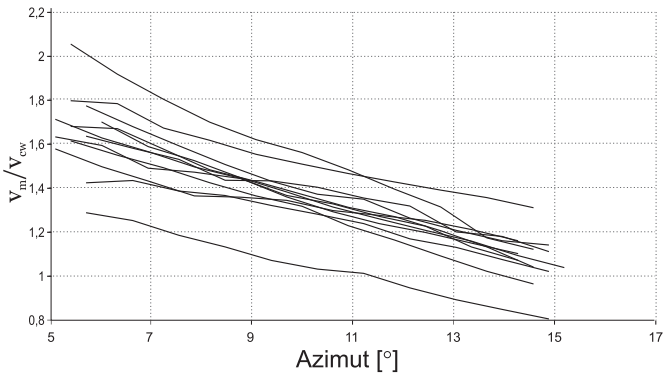
Kako bi se definirali optimalni umjerni parametri, treba pogledati realan izgled krivulja koji se koriste za estimaciju u segmentima I, II i III. Na slici 6.14 prikazane su krivulje dobivene za svaku trojku fototranzistora iz mjerenja sa slike 6.13a u rasponu kuteva segmenta u kojem se koriste.



(a) I v_m/v_{ccw}



(b) II v_{cw}/v_{ccw}



(c) III v_m/v_{cw}

Slika 6.14: Krivulje za estimaciju azimuta u svakom od segmenata dobivene iz izmjerenih vrijednosti na slici 6.13a.

Krivulje u segmentima I i III imaju slične nagibe s različitim odsječcima te je u svrhu umjeravanja optimalno aproksimirati ih pravcima, odnosno polinomom prvog stupnja $ax + b$ s tim da je koeficijent a zajednički za sve trojke fototranzistora, a koeficijent b se određuje za svaku trojku posebno. Za aproksimaciju krivulja u segmentu II koristi se jedan zajednički polinom 4. stupnja. Ukoliko se za svaki koeficijent koristi realan broj s pomičnim zarezom (engl. *float*) veličine 4 byte-a, ukupna količina podataka potrebna za umjerne podatke iznosi $(13+5+13) \cdot 4 = 124$ byte-a. Kako su vrijednosti pojedinih koeficijenata polinoma dobivene za različita osjetila istog reda veličine, ukoliko je potrebno, moguće ih je uz očuvanu točnost, definirati koristeći samo 2 bytea te bi u tom slučaju količina podataka iznosila samo 62 byte-a.

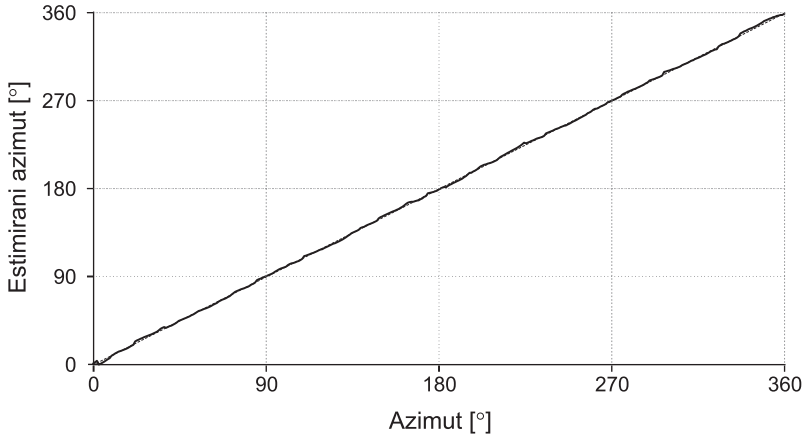
Nakon što su obavljena mjerenja i izračunati odgovarajući umjerni parametri za svako od 16 osjetila, te nakon što je omogućeno čvorovima njihovo korištenje, obavljena su ispitivanja točnosti estimacije azimuta koristeći platformu sa slike 6.12 i opisani protokol. Tipični rezultati prikazani su na slici 6.15.

Na slici 6.16 prikazana je standardna devijacija estimacije azimuta za svako od 16 osjetila. Kao što se može primijetiti iz dobivenih rezultata, standardna devijacija je u većini slučajeva ispod jednog stupnja, što bi prema rezultatima dobivenim u simulatoru trebalo omogućiti vrlo točno estimiranje lokacija. Povećanjem broja parametara, npr. tako da se krivulje za segment I i III definiraju polinomom višeg stupnja i/ili da se i koeficijent a definira za svaku trojku tranzistora, opravdano bi bilo očekivati dodatni napredak.

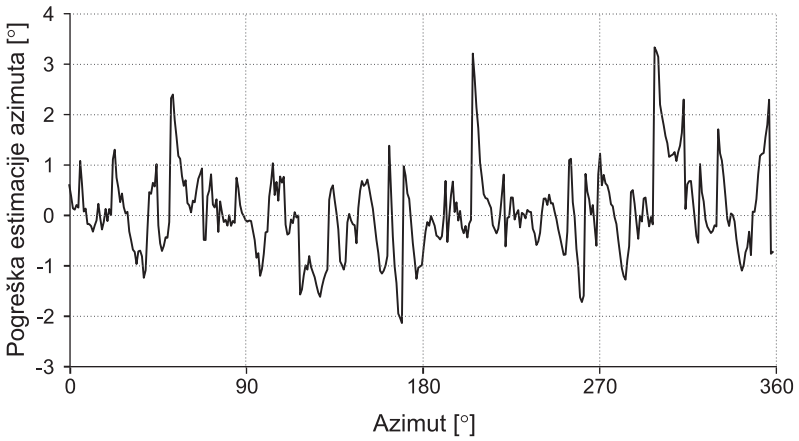
6.3 Implementacija algoritma

U implementaciju algoritma u ispitnoj mreži krenulo se nakon ispitivanja u simulatoru što je omogućilo uvid u očekivane rezultate s aspekta točnosti, te komunikacijske i računalne složenosti. Ispitivanje u simulatoru je dodatno olakšalo rješavanje potencijalnih problema u implementaciji kao što je detekcija završetka pojedinog zadatka ili izbjegavanje zastoja koji se mogu javiti zbog nepredviđenih kašnjenja u komunikaciji. Iz tih razloga se u implementaciji pokušava u što većoj mjeri slijediti verziju algoritma iz simulatora.

Osnovni koncepti, pojedini zadaci, te pripadajuće faze algoritma opisane su u prethodnom poglavlju posebice u tablici 5.1, dok su u ovom dijelu predstavljeni implementacijski detalji. Kao i koncepti iz prošlog poglavlja i implementacijski detalji se u većoj mjeri odnose na obje verzije: u simulatoru i u ispitnoj mreži. Ipak, osim očiglednih razlika, kao što je programski jezik (Python u simulatoru odnosno C/C++ u implementaciji) nije se moglo izbjeći i nekoliko dodatnih promjena koje su bile uvjetovane samom



(a)



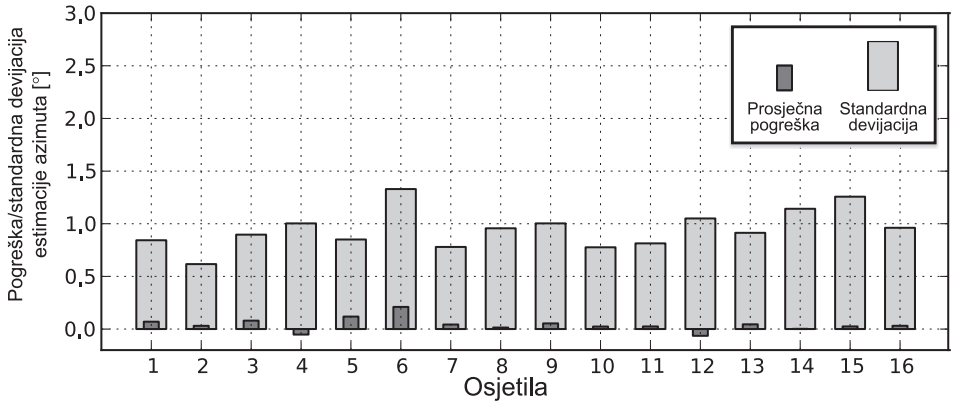
(b)

Slika 6.15: Točnost estimacije azimuta za jedno od osjetila.

platformom.

Najveća promjena je učinjena u drugoj fazi algoritma gdje se u originalnoj inačici algoritma za estimaciju položaja u inicijalnim grozdovima i algoritma za spajanje grozdova koriste metode koje se zasnivaju na dekompoziciji po singularnim vrijednostima – SVD. Algoritam za SVD je implementacijski, računalno i memorijski zahtjevan, stoga je njegovo korištenje u oba zadatka zamijenjeno jednostavnijim, i za implementaciju na mikrokontroleru prikladnijim, mikroimunološkim algoritmom (engl. *MicroImmune Algorithm* – μIA) [12] koji je opisan detaljnije u nastavku poglavlja u dijelu 6.3.2.

Razlike u odnosu na inicijalnu inačicu u simulatoru uključuju još i izostanak podjele po grozdovima te izostanak heuristike za izbjegavanje nedovoljno dobro estimiranih dijelova mreže. Naime, heuristika uključuje izračun pseudoinverza matrice $\boldsymbol{G}$ čije dimenzije su proporcionalne s brojem čvorova u inicijalnom grozdu. Primjerice, za 7 čvorova u inicijalnom grozdu maksimalne dimenzije te matrice su $(2 \cdot (7-1) \cdot 7) \times (3 \cdot 7) = 84 \times 21$ što



Slika 6.16: Prosječna vrijednost i standardna devijacija pogreške estimacije azimuta za pojedina osjetila.

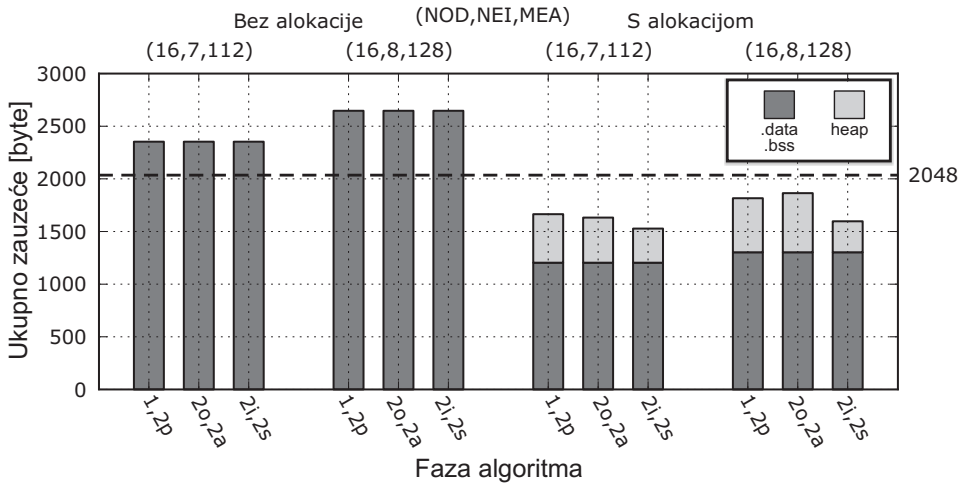
je s obzirom na vrlo ograničenu memoriju odabranog mikrokontrolera, preveliki zahtjev. Naravno, prije same implementacije u simulatoru je ispitana i verzija s navedenim izmjenama koja će poslužiti i za kasniju usporedbu rezultata s onima dobivenim u ispitnoj mreži.

Ispitivanje u simulatoru je dodatno olakšalo donošenje pojedinih odluka. Jedna od takvih odluka je izbor *ConvergecastStitch* algoritma kojim se definira globalni smjer spajanja koordinatnih sustava. Kao što je navedeno u analizi u djelu 5.3.1 prednost *ConvergecastStitch* algoritma u odnosu na *BroadcastStitch*, koji djeluje u suprotnom smjeru, je manja komunikacijska složenost. Mane se prvenstveno odnose na upotrebu u mrežama s puno čvorova u kojim završna spajanja uključuju veliki broj čvorova. Kako se u ispitnoj mreži koristi mali broj čvorova ta mana ne dolazi do izražaja, te je zbog jednostavnosti i brzine odlučeno koristiti *ConvergecastStitch*.

Raspodijeljeni algoritam funkcionira po principu reakcije na događaje (engl. *event based behavior*) kao što su primitak poruke, istjecanje postavljenog alarma te interakcija s entitetima izvan mreže npr. pritisak na tipkalo na čvoru, uključenje čvora i sl. Čvor se u pojedinom trenutku može nalaziti u jednom od konačnog broja definiranih statusa, a algoritmom se definira akcija za svaku kombinaciju događaja i statusa. U prilogu P4 dan je pseudokod algoritma te opis svih statusa i tipova poruka koji se koriste u algoritmu.

Jedan od najvećih izazova implementacije bio je omogućiti izvršavanje algoritma u samo 2kB radne memorije. Kako bi to bilo moguće, napravljena je detaljna analiza potrošnje po pojedinim zadacima koja je dana u prilogu P5. Nakon što su u navedenoj tablici identificirane strukture koje zauzimaju najviše memorije, za dinamičku alokaciju odabrani su redom po fazama sljedeće strukture: `g_aoa`, `g_measurements`, te u zadnjoj fazi `g_destination_cluster` i `g_source_cluster`. Na slici 6.17 prikazano je zauzeće

memorije bez i s dinamičkom alokacijom navedenih struktura.



Slika 6.17: Zauzeće memorije u pojedinim fazama algoritma s i bez korištenja dinamičke alokacije uz parametre (NOD, NEI, MEA) koji redom označavaju maksimalan broj čvorova mreže, maksimalan broj susjeda, te maksimalan broj mjerenja. Isprekidanom crtom za vrijednost 2048 je označena ukupna radna memorija raspoloživa na mikrokontroleru. Razlika između vrha zauzeća i isprekidane crte je prostor koji je na raspolaganju za programski stog.

6.3.1 Faza 1: prikupljanje podataka

Kao što je prikazano u tablici 5.1 osnovni zadatak prve faze algoritma je prikupljanje potrebnih podataka za drugu fazu. Traženi podaci su:

- lista susjeda
- mjerenja azimuta prema susjedima, od susjeda, te između susjeda
- podskup susjeda koji čine susjedi u stablu te njihova uloga (dijete ili roditelj).

Za svaki od navedenih zadataka postoji odgovarajući algoritam, no u implementaciji se iskorištava priroda mjerenja azimuta, te je osmišljen poseban protokol kojim je u jednom obilasku moguće obaviti sve zadatke, odnosno prikupiti sve tražene podatke.

Protokol za detekciju susjeda u grafu mjerenja, mjerenje i estimaciju azimuta, razmjenu rezultata te kreiranje komunikacijskog stabla

Osnovna karakteristika mjerenja azimuta je da čvor kojem se određuje azimut treba vrlo kratko uključiti svoje IR diode na odašiljačkoj strani osjetila. U tom periodu svi čvorovi koji detektiraju zračenje razine koje osigurava pouzdanu estimaciju azimuta, obavljaju mjerenje i estimaciju. Iz tog razloga osnovu novog protokola čini najava uključenja dioda od strane jednog čvora i naknadno mjerenje svih čvorova u komunikacijskom dometu te njihov odgovor. Kako bi mjerenje bilo pouzdano, u pojedinom trenutku isključivo

jedan čvor može imati uključene diode. Stoga je odlučeno za ovu fazu koristiti protokol zasnovan na posebnoj jedinstvenoj poruci, tzv. tokenu. Token je tip poruke koji se u prvoj fazi algoritma prosljeđuje od čvora do čvora duž komunikacijskog stabla u čijem stvaranju i sudjeluje. Samo čvor koji posjeduje token može najaviti mjerenje i uključiti diode. U nastavku teksta pod pojmom token podrazumijeva se i poruka, ali i čvor koji je posljednji primio tu poruku.

Protokol kojim se token prosljeđuje funkcionira na principu raspodijeljenog algoritma za pretraživanje grafa u širinu (engl. *Breadth First Search – BFS*). Prvi vlasnik tokena je čvor inicijator koji ujedno postaje i korijen komunikacijskog stabla. Njemu se token dostavlja ili izvana ili pritiskom na tipkalo koje se nalazi na samom čvoru. On najavljuje svoje uključenje dioda porukom SetThr, a po primitku iste ostali čvorovi izmjere ambijentalno osvjetljenje u IR dijelu spektra. Nakon što je čvor s tokenom ostavio dovoljno vremena za mjerenje ambijentalnog osvjetljenja, uključuje diode i šalje poruku GetAoa na koju svi ostali čvorovi obavljaju još jedno mjerenje, ali ovaj put dodatno obavljaju i estimaciju azimuta najjačeg izvora zračenja odnosno čvora koji posjeduje token. Odmah po obavljenoj estimaciji svim čvorovima u dometu objavljuju rezultat estimacije odnosno azimut tokena u odnosu na njihovu lokaciju i orijentaciju. Čvorovi koji nisu detektirali dovoljno jak izvor zračenja pretpostavljaju da je čvor s tokenom predaleko ili na nepovoljnoj lokaciji za pouzdano mjerenje te uskraćuju svoju objavu. Svi čvorovi osim mjerenja, estimacije i objavljivanja osluškuju i objave ostalih čvorova, a oni čvorovi koji smatraju da su im objavljeni azimuti korisni, iste zapisuju u svoju memoriju. Jedan od kriterija kojim se određuje hoće li čvor zapisati objavljene azimute je odgovor na pitanje je li on sâm objavio estimirani azimut. Ako nije, to znači da mu čvor s tokenom nije susjed, te nema potrebe zapisati ni ostale objavljene azimute čvora s tokenom. Čvor s tokenom dodatno postavlja sve čvorove koji su objavili azimute u listu svojih potencijalnih susjeda. Ukoliko uz objavu estimiranih azimuta čvorovi inidiciraju kako je to njihovo prvo mjerenje, oni postavljaju čvor s tokenom za svog roditelja, a čvor s tokenom njih postavlja u listu svoje djece u stablu.

Sljedeći koncept algoritma pretraživanja u širinu, čvor s tokenom (inicijator, korijen) prosljeđuje token redom svojoj djeci u stablu. Djeca po primitku tokena obave identičan, prethodno opisani, niz akcija: pošalju poruku SetThr, uključe diode te pošalju poruku GetAoa te iz objava saznaju svoju djecu u stablu, potencijalne susjede i svoj azimut u njihovim koordinatnim sustavima. Nakon toga vraćaju token korijenu koji zatim pokreće drugu iteraciju u kojoj njegova djeca prosljeđuju token svojoj djeci, ukoliko ih imaju. Ako neki čvor nakon obavljanja protokola mjerenja sazna kako nema djece (list stabla), to dojavljuje svom roditelju koji mu više neće prosljeđivati token, a sam prelazi u status DONEP1. Ukoliko unutarnji čvor stabla (čvor koji nije ni korijen ni list) sazna da su mu

sva djeca u statusu DONEP1, sam može prijeći u taj status te javiti isto svom roditelju. Prva faza završava kada korijen sazna od sve svoje djece da su u statusu DONEP1. U tom trenutku sa sigurnošću može zaključiti kako su svi čvorovi mreže u statusu DONEP1 i imaju sve podatke potrebne za izvršavanje druge faze algoritma.

Problem pristupa mediju

Najveći izazov u implementaciji prve faze i razlika u odnosu na verziju iz simulatora bila je implementacija objava rezultata estimacije. Naime algoritam u simulatoru funkcionira uz pretpostavku kako će svaka poslana poruka uz određeno nepredviđeno, ali konačno kašnjenje stići do svog odredišta. Ono što se ne specificira je metoda kojom će to biti ostvareno, točnije protokol pristupa mediju (engl. *Media Access Control – MAC*). U ovom konkretnom slučaju problem je potencijalna istovremena objava rezultata estimacije te posljedično ometanje. Implementirano rješenje ovog problema koristi jedinstveni vremenski odmak prije same objave. Trajanje pojedine objave koja osim zaglavlja i rezultata estimacije, uključuje još samo indikaciju je li čvor čiji azimut se objavljuje roditelj, uz izabranu brzinu prijenosa kraći je od 3 ms. Čvorovi u ispitnoj mreži posjeduju jedinstvene ID brojeve, te je konačni vremenski odmak za objavu svakog od njih jednak umnošku njihovog ID broja i dovoljno dugog odmaka, koji mora biti veći od navedenih 3ms tako da ne dođe do ometanja. Bitno je naglasiti kako čvor za sam odmak ne smije koristiti blokirajuće čekanje (engl. *delay*) jer prije i nakon objave mora osluškivati objave drugih čvorova i ukoliko je potrebno zapisivati ih. Iz tog razloga čvor nakon estimacije priprema poruku, ali ne daje zahtjev za slanje već postavlja alarm čijim istjecanjem se pokreće odašiljanje. U međuvremenu prijamnik je slobodan primati objave drugih čvorova.

6.3.2 Faza 2: određivanje položaja u inicijalnim grozdovima i protokol spajanja

Kao što je već napomenuto, algoritam RAST za estimaciju položaja u inicijalnim grozdovima zamijenjen je tzv. mikroimunološkim algoritmom (μI), dok se za spajanje koordinatnih sustava koristi raspodjeljeni algoritam *ConvergecastStitch*. Detalji implementacije oba algoritma dani su u nastavku kao i detalji komunikacijskog protokola kojim se osigurava pouzdana dostava zahtjeva za spajanjem.

Mikroimunološki algoritam

Mikroimunološki algoritam je optimizacijski algoritam razvijen prilikom rješavanja problema inverzne kinematike redundantnog robota [12]. Nastao je pojednostavljivanjem

mikrogenetičkog algoritma [73], odnosno ukidanjem operatora križanja. Ovom pojednostavljenju algoritam duguje naziv jer imunološki algoritmi su klasa evolucijskih algoritama koja prilikom evolucije koristi samo mutaciju. Kako bi se osigurala konvergencija, algoritam koristi operator kontrakcije prostora koji smanjuje prostor pretraživanja oko najbolje jedinke. Nakon svake iteracije, formira se hiperkugla radijusa koji se smanjuje geometrijskim redom. Veličina prostora pretraživanja koristi se kao kriterij zaustavljanja algoritma odnosno, nakon što radijus prostora pretraživanja postane premali za pronalazak boljeg rješenja, algoritam staje s radom. Parametri algoritma su broj populacija koje se evaluiraju u jednoj iteraciji L, te faktor kontrakcije prostora λ.

Algoritam se koristi u dva koraka: (1) za estimaciju nepoznatih orijentacija koje su preduvjet izračuna azimuta usklađenih s lokalnim koordinatnim sustavom te (2) za određivanje lokacija u inicijalnim grozdovima iz tako dobivenih azimuta. U daljnjem tekstu pretpostavlja se estimacija orijentacija i lokacija u lokalnom koordinatnom sustavu čvora x. U prvom slučaju estimacije orijentacija čvorova susjeda čvora x kriterijska funkcija poprima sljedeći oblik:

$$\max_{\forall(r,t)\in\mathcal{E}_{N(x)}} (\phi_{tr} - \alpha_r - (\phi_{rt} - \alpha_t) + \pi) \tag{6.1}$$

pri čemu je s $\mathcal{E}_{N(x)}$ označen skup bridova odnosno estimiranih azimuta u susjedstvu čvora x. ϕ_{rt} i ϕ_{tr} označavaju estimirane azimute čvorova r i t u međusobnim lokalnim koordinatnim sustavima, a čije orijentacije α_r i α_t se estimiraju. Kao što je spomenuto, mikroimunološki algoritam rješava optimizacijski problem u kojem se kriterijska funkcija (6.1) minimizira s obzirom na orijentacije $\alpha_r, \forall r \in N(x)$.

Nakon estimacije orijentacija računaju se azimuti usklađeni s lokalnim koordinatnim sustavom čvora x:

$$\theta_{rt} = \phi_{rt} + \alpha_r \tag{6.2}$$

te potom i odgovarajući jedinični vektori u_{rt} koji odgovaraju onima iz jednadžbe (P.1):

$$\boldsymbol{u}_{rt} = \begin{bmatrix} \sin\theta_{rt} \\ \\ -\cos\theta_{rt} \end{bmatrix} \tag{6.3}$$

S obzirom da se želi zamjeniti metoda iz priloga P1, kriterijska funkcija za populaciju uzima vektor $p = [p_{x_1}, p_{y_1}, p_{x_2}, p_{y_2}, \ldots, p_{x_n}, p_{y_n}]$ koji se sastoji od koordinata čvorova u susjedstvu čvora x. Za danu populaciju odnosno formaciju najprije se izračuna centroid te potom obavi translacija tako da ishodište nove formacije $\bar{p}$ bude u izračunatom centroidu.

Kriterijska funkcija koja odgovara jednadžbi (P.11) jednaka je:

$$J' = \sum_{(r,t)\in\mathcal{M}} \left(u_{rt}^T(p_t - p_r)\right)^2 \tag{6.4}$$

Kako je rješenje dobiveno koristeći SVD takvo da je $||x|| = 1$, za konačnu vrijednost kriterijske funkcije J potrebno je obaviti normalizaciju:

$$J = \frac{J'}{||\overline{p}||^2} \tag{6.5}$$

pri čemu je s $||\overline{p}||^2$ označena norma vektora odnosno:

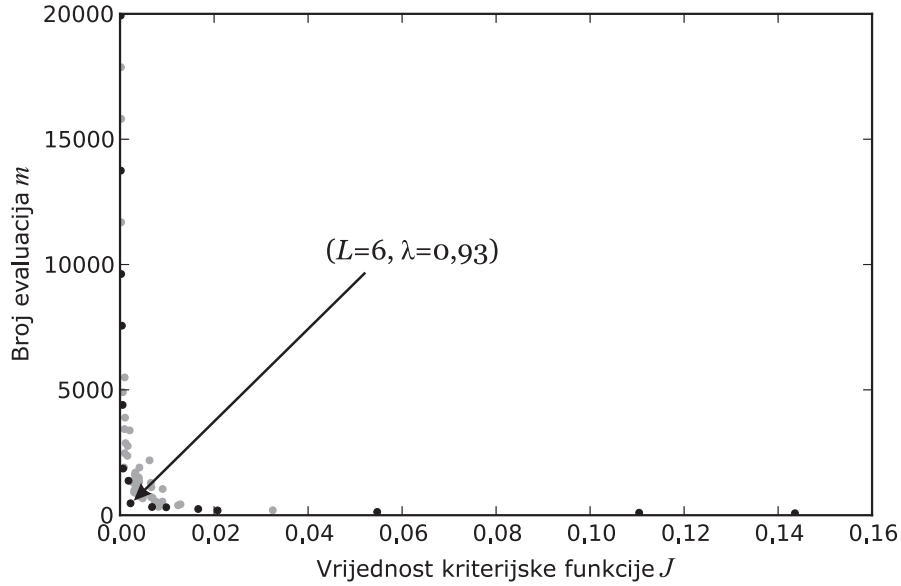
$$||\overline{p}||^2 = \sum_n \overline{p}^2. \tag{6.6}$$

Bez provedene normalizacije, minimiziranje kriterijske funkcije, umjesto željenog pomaka populacije bliže pravim lokacijama, moglo bi biti obavljeno i jednostavnim smanjenjem udaljenosti između estimiranih lokacija što bi rezultiralo nedovoljno kvalitetnom estimacijom.

U svrhu odabira parametara algoritma L i λ obavljen je niz simulacija μI algoritma primjenjenog na estimaciju položaja u formacijama s 3 do 6 čvorova u kojima su korištene sve kombinacije parametara $L \in \{3, 6, \ldots, 30\}$ i $\lambda \in \{0{,}84, 0{,}87, \ldots, 0{,}99\}$. Kao kriterij izbora jedne od kombinacija s Pareto fronte prikazane na slici 6.18 odabran je minimalni broj evaluacija kriterijske funkcije m, a kojim se dobiva dovoljno mala vrijednost kriterijske funkcije J. Odabrana kombinacija koja se koristi u daljnjim simulacijama i implementaciji je $L = 6$ i $\lambda = 0{,}93$.

Kako bi se ispitalo funkcioniranje μI algoritma kao zamjene za algoritam RAST, najprije je izmijenjena verzija u simulatoru, te je provjereno koliko zamjena utječe na kvalitetu rezultata. Za simulaciju su odabrane 24 različite mreže s 3 do 6 čvorova i standardnom devijacijom pogreške mjerenja azimuta od 5°, te je obavljena estimacija lokacija koristeći RAST i μI algoritam uz parametre $L = 6$ i $\lambda = 0{,}93$. Na slici 6.19 prikazan je korijen srednje kvadratne pogreške estimacije lokacija i kao što se vidi iz rezultata, μI algoritam u većini slučajeva uspijeva pronaći rješenje koje je kvalitetom usporedivo sa SVD inačicom.

U slučajevima kada kvaliteta rješenja nije zadovoljavajuća, vrijednost kriterijske funkcije J za konačno rješenje je veća od očekivane za danu kvalitetu mjerenja azimuta. Stoga se u cilju izbjegavanja nekvalitetnih rješenja odnosno lokalnih minimuma, u konačnu inačicu uvelo dva nova parametra. To su prag za vrijednost kriterijske funkcije $J_{\max}$ te J_{iter}, odnosno maksimalni broj ponavljanja μI algoritma u cilju pronalaska rješenja koji bi bio



Slika 6.18: Vrijednosti kriterijske funkcije za konačno rješenje uz različite kombinacije parametara L i λ odnosno odgovarajućeg broja iteracija $m = 1 + L \log_\lambda d$ uz $d = 10^{-3}$. Vrijednosti na Pareto fronti označene su crnom bojom.

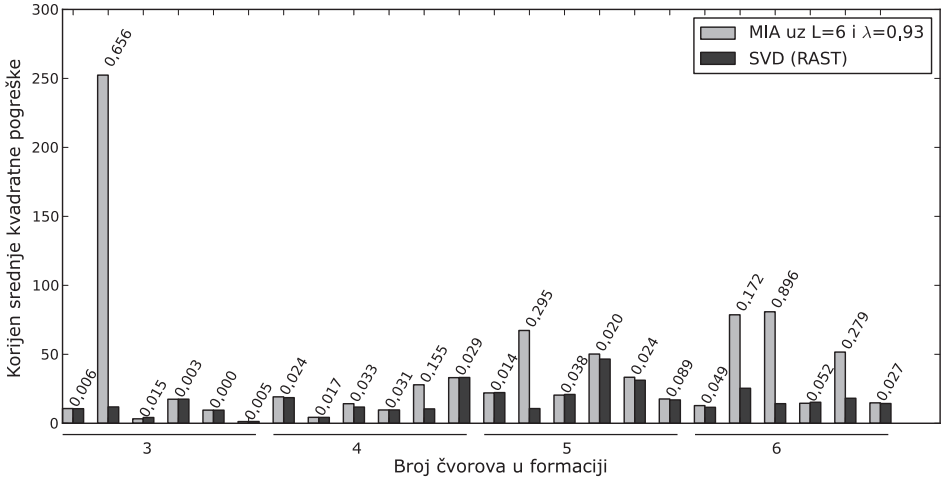
ispod zadanog praga.

Kako bi se omogućilo dodatno ispitivanje te uvid u ponašanje μI algoritma, napisana je programska knjižnica u C-u koja se koristi u konačnoj implementaciji te je omogućeno njeno izravno korištenje u simulatoru. S obzirom da se radi o stohastičkom algoritmu te kako bi se omogućila izravna usporedba, implementirano je dijeljenje istog generatora slučajnih brojeva. Testovi su pokazali kako algoritam u Pythonu i verzija u C-u daju identične rezultate.

Convergecaststitch

Kao što je već spomenuto u dijelu 5.3.1, najvažnija karakteristika *ConvergecastStitch* algorima je globalni poredak spajanja koordinatnih sustava od listova prema korijenu komunikacijskog stabla. Algoritam se zasniva na raspodijeljenom algoritmu *Convergecast* opisanom u [62], a pseudokod je dan u prilogu P4. Jedan od preduvjeta je korijensko stablo koje je osigurano u 1. fazi algoritma, a algoritam se sastoji od sljedećih pravila:

1. Korijen stabla odašilje poruku StartStitch svim ostalim čvorovima, opcionalno u više komunikacijskih skokova, te se postavlja u status STITCHING.
2. Nakon primitka poruke StartStitch, u status STITCHING prelaze i svi unutarnji čvorovi stabla, dok svaki list stabla šalje svojem roditelju poruku LetUsStitch u kojoj šalje lokacije čvorova u svom inicijalnom grozdu. Kako je time njegov zadatak završen, list postavlja svoj status u WAITLOC.
3. Roditelj, nakon što primi poruku LetUsStitch od svog djeteta, spaja dva koordinatna



Slika 6.19: Usporedba kvalitete estimacije lokacije μI algoritma s algoritmom RAST zasnovanim na SVD-u. Iznad stupca koji označava μI algoritam upisana je vrijednost kriterijske funkcije J za dobiveno rješenje.

sustava metodom najmanjih kvadrata opisanom u prilogu P2.

4. Roditelj, nakon što primi LetUsStitch poruke od sve svoje djece, šalje istu poruku svom roditelju.
5. Algoritam završava kada korijen primi LetUsStitch poruku od sve svoje djece. Korijen u tom trenutku ima konačne estimirane lokacije svih čvorova mreže i to unutar svog koordinatnog sustava, koje potom može i ne mora podijeliti s ostalim čvorovima u mreži.

Jedan od problema koji je bilo potrebno riješiti prilikom implementacije ovog dijela algoritma je pouzdano slanje LetUsStitch poruka od djece prema roditeljima. U ovom slučaju nije se moglo koristiti metodu jedinstvenog trajanja odgode slanja kao što je bio slučaj kod odgovaranja na poruku GetAoa zato jer LetUsStitch poruke nisu sinkronizirane, odnosno njihovo slanje nije vremenski određeno. U ovom slučaju pribjeglo se korištenju opcije zahtjevanja potvrde primitka koja je već opisana u djelu 6.1.3. U slučaju da odgovor potvrde izostane, poruka se šalje ponovno nakon određenog slučajno odabranog intervala odgode. Kako su čvorovi koji šalju LetUsStitch poruku završili sa svim aktivnostima bilo je moguće umjesto postavljanja alarma koristiti jednostavno blokirajuće čekanje.

6.4 Rezultati eksperimenata

Za eksperimentalnu verifikaciju postavljena je mreža koja se sastoji od ukupno 14 čvorova na prostoru 3x3 m u dvije različite topologije prikazane na slici 6.20. Točne lokacije i orijentacije čvorova ispitne mreže ručno su izmjerene koristeći metar i kutomjer, a kako

su dimenzije čvora 11x4 cm za točnu lokaciju odabrano je središte osjetila za mjerenje azimuta.

S obzirom na dimenzije mreže, komunikacijski domet omogućava međusobnu komunikaciju svih parova čvorova. S druge strane, domet osjetila omogućava estimaciju azimuta samo između pojedinih čvorova. Iako bridovi u grafovima koji su prikazani u nastavku poglavlja predstavljaju mogućnost estimacije azimuta, zbog karakteristika algoritma i komunikacija je također, uz jednu iznimku, ograničena samo na parove čvorova koji su uspješno obavili estimaciju azimuta. Iznimku učinjenu zbog pojednostavljenja algoritma čine po jedna poruka DoIniLoc i StartStitch koje korijen odašilje svim čvorovima kod objave početka druge faze, no prva se u konačnoj verziji može slati primjerice *Broadcast* algoritmom, a druga čak i u potpunosti izostaviti.

(a) Konveksna topologija (b) Nekonveksna topologija

Slika 6.20: Fotografije ispitnih mreža za dvije različite topologije.

6.4.1 Metoda provođenja eksperimenata

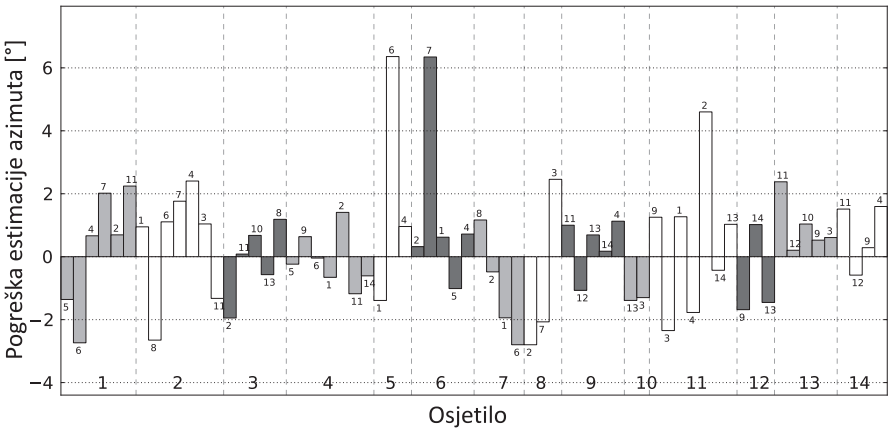
Eksperiment započinje ili slanjem poruke Token čvoru koji se odabere kao korijen ili, alternativno, pritiskom na tipkalo koje se nalazi na samom čvoru. To je jedini događaj čiji uzrok se nalazi izvan same mreže, a algoritam se dalje izvršava prema opisanom protokolu odnosno sve daljnje korake čvorovi provode potpuno autonomno. Za analizu algoritma tijekom njegovog izvršavanja koristi se poseban čvor priključen na računalo s, u protokolu posebno definiranom, adresom 31 koji pasivno osluškuje sve poruke između čvorova te ih u prikladnom formatu ispisuje na ekran odnosno u datoteku. Također,

dodatno je omogućena i vizualna inspekcija korištenjem svijetlećih dioda na čvorovima, a koje indiciraju njihov status u pojedinom trenutku izvršavanja algoritma.

Prije eksperimenta moguće je korištenjem čvora priključenog na računalo, posebno definiranim tipovima poruka konfigurirati čvorove. Konfiguracija uključuje eventualnu promjenu identifikacijskog broja i grupe, te parametara mikroimunološkog algoritma i komunikacijskog protokola. Nakon eksperimenta na isti način se može učitavati memorija čvorova, a sadržaj se u formi prethodno navedenih zapisa u prikladnom formatu ispisuje na ekranu odnosno u datoteku. Analizom datoteke sa zapisima moguće je saznati relevantne informacije o provedenom eksperimentu odnosno iscrtati odgovarajuće grafičke prikaze. U tu svrhu na raspolaganju su i alati programskog simulatora opisanog u poglavlju 7.

6.4.2 Konveksna topologija

Analiza eksperimenta provedenog korištenjem konveksne topologije sa slike 6.20a započinje analizom pogreške estimacije azimuta u prvoj fazi algoritma. Standardna devijacija pogreške estimacije azimuta dobivena je iz podataka prikazanih na slici 6.21 i iznosi 1,839°. To uključuje i pogrešku ručnog očitanja azimuta čvorova pri čemu se očitana vrijednost zaokruživala na cijeli broj.

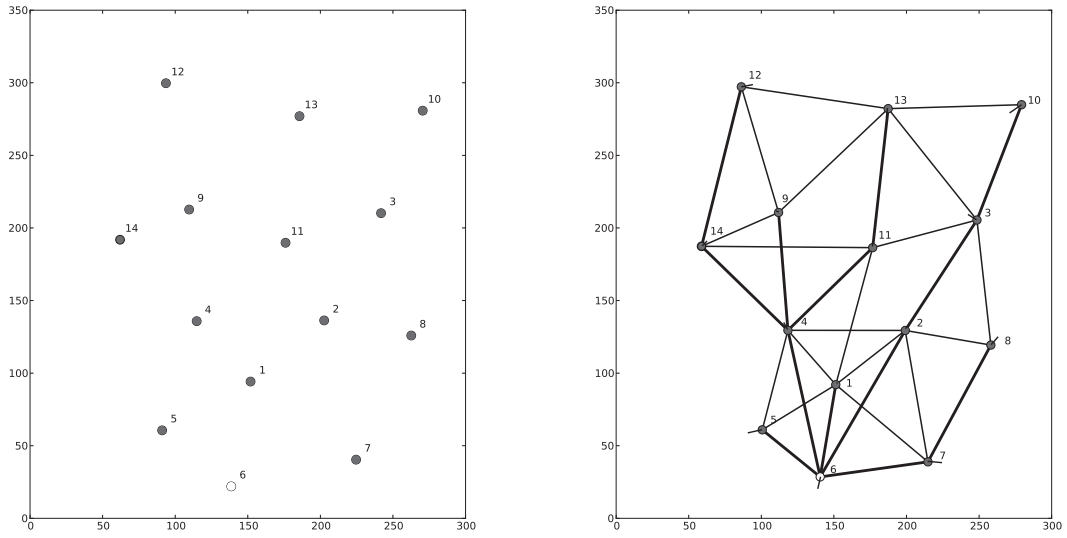


Slika 6.21: Pogreške estimacije azimuta učinjene za vrijeme izvršavanja algoritma. Segmenti označeni brojevima na x osi predstavljaju estimacije azimuta od strane naznačenog čvora, dok brojevi uz prikaz pogreške označavaju ID čvora čiji se azimut estimirao. Standardna devijacija pogreške estimacije azimuta iznosi 1,8°.

Iz tako dobivene standardne devijacije mjerenja azimuta, točnih lokacija čvorova i definiranih bridova grafa računa se Cramér-Rao limit koji za navedenu topologiju iznosi 2,7 cm. Na slici 6.22 prikazan je rezultat estimacije položaja za mrežu s konveksnom topologijom. Korijen srednje kvadratne pogreške iznosi 6,9 cm, što je u usporedbi s

veličinom čvora od 11x4 cm dobar rezultat, no u usporedbi s Cramér-Rao limitom, ne i optimalan rezultat.

Daljnja analiza pogrešaka estimacije azimuta sa slike 6.21 pokazuje kako su pogreške za pojedina mjerenja zamjetno veće od onih na slici 6.16 dobivenih ispitivanjem osjetila. Kao što se vidi iz rezultata, najveće pogreške učinjene kod estimacija azimuta koja uključuju čvor 6. Taj čvor se nalazi samo 22 cm od zida, pa je pretpostavka kako je uzrok većim pogreškama estimacije azimuta povećana razina reflektiranog infracrvenog signala. Ta pretpostavka je i potvrđena prislanjanjem slabo reflektivne površine uz zid i naknadnim mjerenjima te estimacijom s očekivanom smanjenom razinom pogreške.

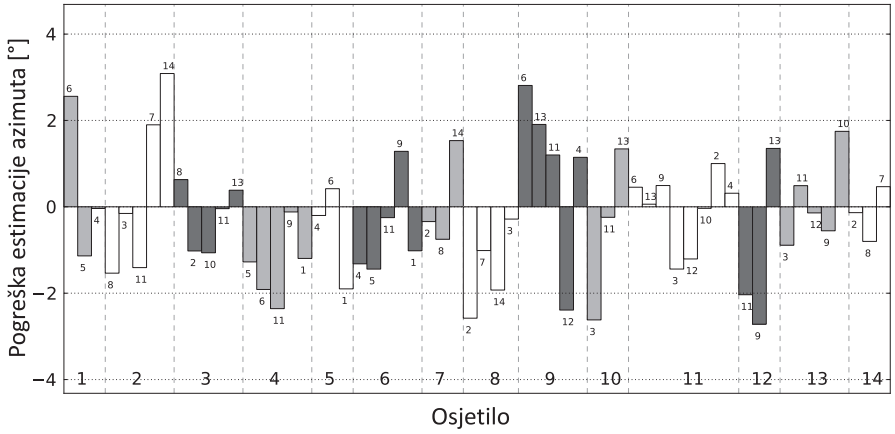


Slika 6.22: Konveksna topologija: lokacije čvorova, označenih s •, dobivene ručnim mjerenjem s lijeve strane, te s desne strane topologija mreže nakon izvršenja algoritma pri čemu su pogreške označene crtama koje počinju u prikazanoj estimiranoj lokaciji, a završavaju u koordinatama točne lokacije. Komunikacijsko stablo označeno je debljim bridovima, a korijen s ∘.

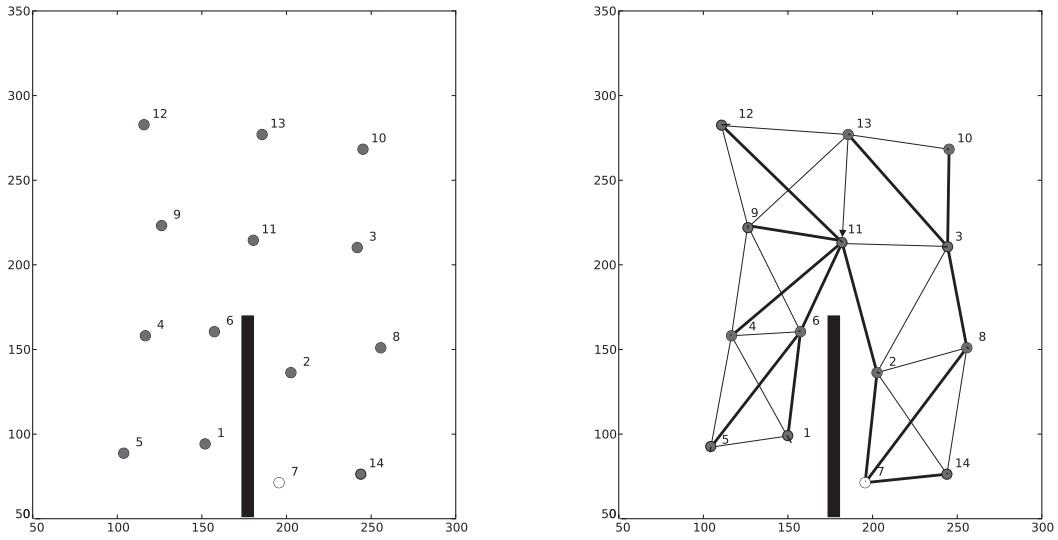
6.4.3 Nekonveksna topologija

Slična analiza provedena je i za nekonveksnu topologiju. Standardna devijacija estimacije azimuta dobivena iz podataka prikazanih na slici 6.23 je 1,38°, dok Cramér-Rao limit iznosi 3,7 cm.

Korijen srednje kvadratne pogreške estimacije lokacije sa slike 6.24 u ovom slučaju iznosi samo 2,6 cm što je manje i od Cramér-Rao limita, te se može zaključiti kako je algoritam vrlo uspješno iskoristio sva rapoloživa mjerenja. Ovakav rezultat, iako je ispod limita nije neočekivan s obzirom na stohastičku prirodu mikroimunološkog algoritma i distribucije pogreške mjerenja azimuta, no zasigurno spada u red najboljih mogućih ishoda za navedenu mrežu.



Slika 6.23: Pogreške estimacije azimuta učinjene za vrijeme izvršavanja algoritma. Segmenti označeni brojevima na x osi predstavljaju mjerenja azimuta od strane naznačenog čvora, dok brojevi uz prikaz pogreške označavaju ID čvora čiji se azimut estimirao. Standardna devijacija iznosi 1,38°.



Slika 6.24: Nekonveksna topologija: lokacije čvorova, označenih s •, dobivene ručnim mjerenjem s lijeve strane, te s desne strane topologija mreže nakon izvršenja algoritma pri čemu su pogreške označene crtama koje počinju u prikazanoj estimiranoj lokaciji, a završavaju u koordinatama točne lokacije. Komunikacijsko stablo označeno je debljim bridovima, a korijen s ○.

Jedan od uzroka boljeg apsolutnog iznosa rezultata u usporedbi s konveksnom topologijom je i manja pogreška estimacije azimuta. Naime, pri postavljanju ovog eksperimenta čvorovi su dodatno odmaknuti od zidova te je skoro u potpunosti otklonjen problem refleksija signala od zidova. U toj maloj, ali bitnoj razlici, očituje se osjetljivost estimacije lokacija na pojedine pogreške estimacije azimuta i njihovu propagaciju, posebice uz vrlo malu povezanost čvorova kao što je slučaj u provedenim eksperimentima.

U slučaju mreže s nekonveksnom topologijom spajanja koordinatnih sustava algori-

tam obavlja duž stabla s većom visinom. Usprkos tome, algoritam i u tom slučaju ne unosi pogrešku. Razlog tome je izostanak većih pogrešaka estimacije položaja u inicijalnim grozdovima, ali i metoda spajanja koja ne unosi dodatnu pogrešku.

6.4.4 Verifikacija - usporedba s rezultatima simulacija

Osim verifikacije algoritma dodatni cilj je i verifikacija programskog simulatora. Simulator je izrađen u sklopu istraživanja, a njegov detaljniji opis je dan u poglavlju 7. Usporedba rezultata u simulatoru i u samom eksperimentu obavljena je s aspekta točnosti estimacije lokacija. U simulatoru je kreirana mreža s točnim lokacijama te je postavljena standardna devijacija estimacije azimuta utvrđena u samom eksperimentu. Zatim je simulacija pokrenuta 30 puta, te je zapisivan konačni iznos korijena srednje kvadratne pogreške estimacije lokacija.

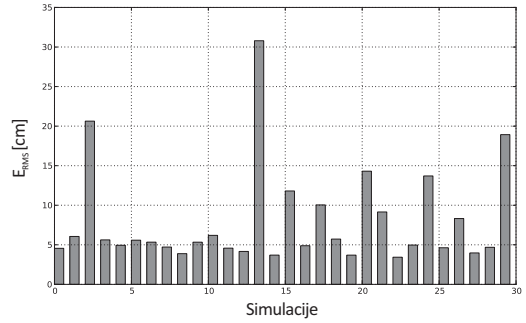
Razlike između pojedinih simulacija su:

- Estimirani azimuti. Iako je standardna devijacija estimacija azimuta ista, konkretna mjerenja se razlikuju.
- Čvor inicijator, odnosno korijen komunikacijskog stabla, te posljedično i samo stablo, odnosno lokalni i globalni poredak spajanja.
- Slučajno generirana rješenja prilikom izvršavanja mikroimunološkog algoritma.

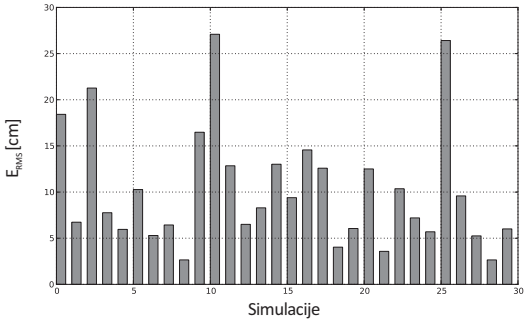
Rezultati su prikazani na slici 6.25. Kao što se može uočiti, pogreška estimacije ima širok raspon vrijednosti. Razlog tome je prvenstveno mala povezanost čvorova, te pogreške u pojedinim mjerenjima azimuta koje mogu imati veliki utjecaj na konačni rezultat. Dodatno, pogreške mjerenja azimuta su modelirane normalnom razdiobom što pruža mogućnost velikog odstupanja za pojedinačna mjerenja, a što nije slučaj kod osjetila koje se koristi u eksperimentu. Neovisno o tome, ako se promatra prosječna vrijednost i standardna devijacija za provedene simulacije, tada rezultat dobiven u eksperimentu odgovara vrijednostima dobivenim u programskom simulatoru. Za nekonveksnu topologiju dodatno se može primijetiti kako dobiveni rezultat odgovara najboljim rezultatima postignutim u simulatoru.

6.4.5 Komunikacijsko opterećenje

Komunikacijsko opterećenje, a posljedično i veliki dio energetskog opterećenja čvora ovisi o količini odaslanih i primljenih podataka. Prema podacima iz dokumentacije za korišteni radio modul RFM12B, potrošnja energije, odnosno tipična jakost struje iznosi između 11 i 15 mA prilikom prijama, te između 15 i 28 mA prilikom odašiljanja. Snaga prilikom odašiljanja može prema potrebi biti dodatno smanjena, no ta opcija u algoritmu nije iskorištena.



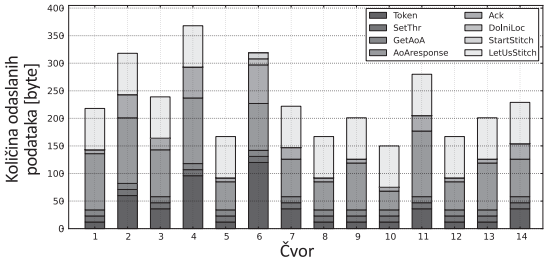
(a) $\overline{E}_{\mathrm{RMS}} = 7.9$ cm, $\sigma_{\mathrm{RMS}} = 6.1$ cm.



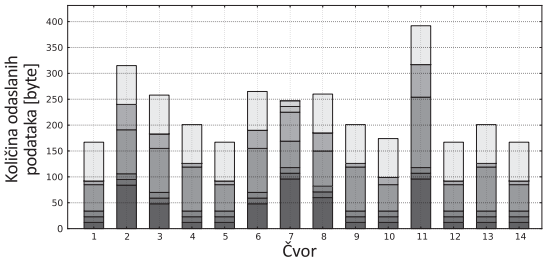
(b) $\overline{E}_{\mathrm{RMS}} = 10.1$ cm, $\sigma_{\mathrm{RMS}} = 6.3$ cm.

Slika 6.25: Rezultati simulacije provedeni za ispitnu mrežu s (a) konveksnom (b) nekonveksnom topologijom, te sa standardnom devijacijom estimacije azimuta koja odgovara vrijednostima dobivenim u eksperimentu. Uz rezultate su naznačene prosječne vrijednosti pogrešaka estimacije, te njihova standardna devijacija.

Na slici 6.26 i 6.27 prikazana je količina primljenih i odaslanih podataka po čvoru i prema tipu poruke. Vidljivo je kako je količina primljenih podataka primjetno veća od odaslanih. Razlog tome je prvenstveno osluškivanje odaslanih estimacija azimuta u porukama AoAresponse, a koje čine najveći udio primljenih poruka.



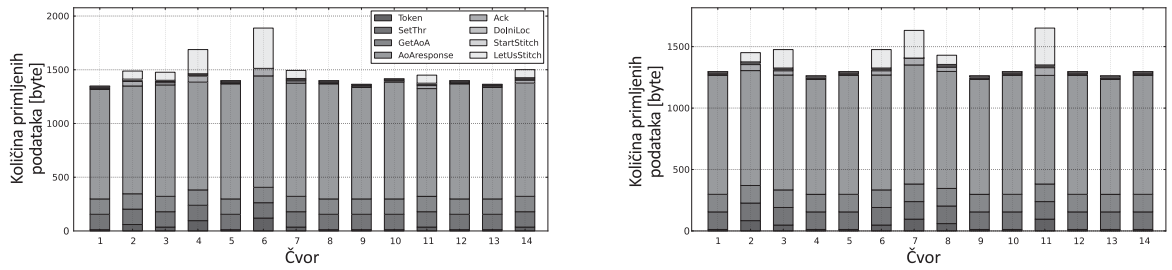
(a) Konveksna topologija



(b) Nekonveksna topologija

Slika 6.26: Količina odaslanih podataka prema tipu poruke i čvoru. Količina podataka za LetUsStitch poruke ovisi o korištenom protokolu, pa je prikazana najmanja moguća količina odnosno jedna poruka po čvoru za sve čvorove osim korijena koji jedini ne šalje tu poruku.

Iako se na svim slikama jasno uočava karakteristika raspodijeljenog algoritma koji ravnomjerno opterećuje sve čvorove mreže, neizbježno je nešto veće opterećenje korijena i čvorova s više djece u komunikacijskom stablu. U tom nadvišenju opterećenja prevladavaju poruke Token u prvoj fazi i kod slanja i kod primanja, te LetUsStitch kod primanja. Ukoliko je potrebno, ovakav trend neravnomjernog opterećenja može se kompenzirati u sljedećim iteracijama izvršavanja algoritma jednostavnim reizborom korijena i zamjenom roditelja i djeteta svih čvorova na putu od starog prema novom korijenu. Druga opcija je rekreiranje stabla s novoodabranim inicijatorom, što je prikladno u slučajevima kada je kreiranje stabla sastavni dio određenog zadatka i ne zahtijeva dodatno opterećenje, kao što je to slučaj u prvoj fazi opisanog algoritma.



(a) Konveksna topologija

(b) Nekonveksna topologija

Slika 6.27: Količina primljenih podataka prema tipu poruke i čvoru. Količina podataka za LetUsStitch poruke ovisi o korištenom protokolu, pa je prikazana najmanja moguća količina odnosno jedna poruka po djetetu.

6.4.6 Vremenska složenost

Na slici 6.28 prikazan je vremenski dijagram izvršavanja algoritma s aspekta poruka koje je primio čvor priključen na računalo.

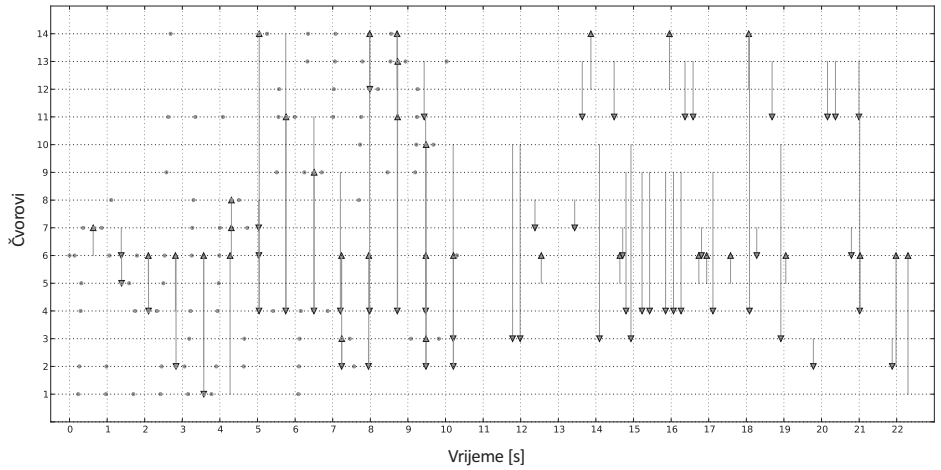
Kao što se vidi iz slike, prva faza algoritma ima konstantno vrijeme trajanja koje ovisi isključivo o broju čvorova, bez obzira na topologiju mreže, a iznosi nešto više od 10 s. Vrijeme prve faze određeno je ponajviše duljinom mjerenja koje iznosi oko 300 ms: 200 ms za mjerenje ambijentalnog osvjetljenja i dodatnih 100 ms za mjerenje zračenja s čvora čiji se azimut mjeri. Samo prosljeđivanje tokena duž stabla traje vrlo kratko te ne sudjeluje značajno u ukupnom vremenu.

Trajanje druge faze ovisi ponajviše o trajanju mikroimunološkog algoritma i broju iteracija pozivanja algoritma kako bi se postigao rezultat ispod zadanog praga $J_{\max}$. Trajanje jednog poziva mikroimunološkog algoritma, iako ima deterministički određen broj koraka koji ovisi o parametrima L i λ, ovisi i o trajanju jedne evaluacije kriterijske funkcije što pak ovisi o broju čvorova, odnosno broju mjerenja obavljenih u inicijalnom grozdu. U konkretnim primjerima, algoritam se izvršavao nešto dulje od 20 sekundi, no općenito u provedenim ekspermentima je varirao od 16 do 30 s.

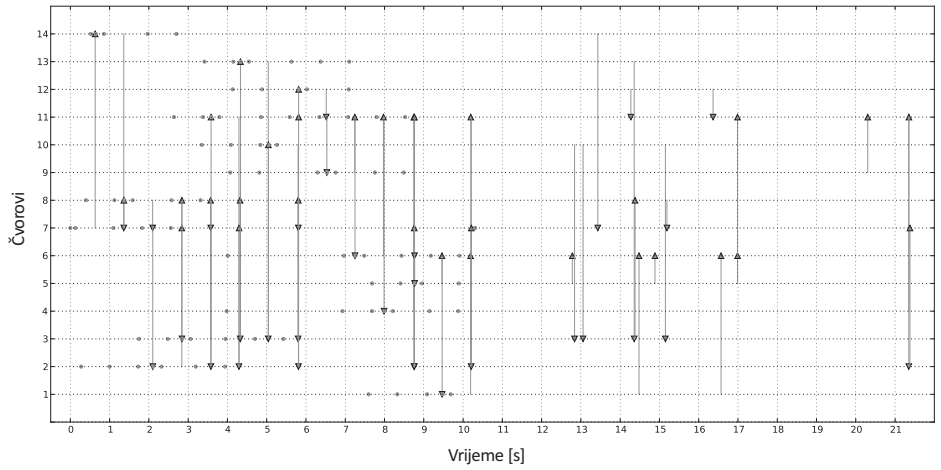
Iako optimizacija trajanja algoritma nije u fokusu istraživanja, uočeno je nekoliko mjesta na kojima bi se moglo intervenirati u cilju ubrzanja izvršavanja. To su redom: optimizacija trajanja mjerenja, npr. reducirati broj mjerenja ambijentalnog osvjetljenja, zatim prelazak na aritmetiku s nepomičnim zarezom i ograničenje prostora pretraživanja, te shodno tome i dodatno podešavanje parametara L i λ u cilju smanjenja ukupnog broja koraka mikroimunološkog algoritma.

6.4.7 Uočeni nedostaci implementacije

Zbog male količine radne memorije, veličine struktura u kojima se zapisuju mjerenja su ograničena, a kao što je prikazano u prilogu P5 njihova veličina ponajviše ovisi o mak-



(a) Konveksna topologija



(b) Nekonveksna topologija

Slika 6.28: Prikaz poruka u vremenskom kontekstu za provedene eksperimente. Strelicama je označen smjer poruke, a točkama su označene poruke namijenjene svim čvorovima, kao što je primjerice AoAresponse. Radi preglednosti poruke potvrde primitka su izostavljene.

simalnom dopuštenom broju susjeda. U inačici koja je ispitivana, maksimalni mogući broj susjeda ograničen je na 7, stoga su raspored čvorova u ispitnoj mreži i domet osjetila posebno odabrani kako bi se poštovalo to ograničenje. Negativna strana navedenog ograničenja je, trenutno, prilično smanjena mogućnost prilagodbe algoritma različitim topologijama. Kako bi se to ograničenje primjene izbjeglo moguće je koristiti raspodijeljenu selekciju susjeda odnosno bridova grafa po principu njihovog utjecaja na krutost cijele mreže, no to je problem koji zaslužuje posebno istraživanje. U sklopu ovog istraživanja osnovni cilj implementacije je verifikacija funkcioniranja algoritma, te s obzirom da su navedena ograničenja isključiva posljedica nedostatka memorije taj cilj nije upitan. Štoviše, navedeno ograničenje je u suprotnosti s principom prema kojem više susjeda omogućava bolju estimaciju lokacije, pa dolazi do izražaja sposobnost algoritma da

omogući točnu estimaciju i u uvjetima male povezanosti.

Poglavlje 7

Pymote: simulator raspodijeljenih algoritama

Kako bi smo uspješno vrednovali raspodijeljene algoritme potrebno ih je adekvatno simulirati i usporediti s recentnim rješenjima u području. Iako se ovaj zadatak može učiniti jednostavan, u praksi postoji nekoliko problematičnih detalja čijem rješavanju se treba pristupiti pažljivo. Postojeća simulacijska okruženja često zahtijevaju definiciju velikog broja parametara na nižim razinama komunikacijskog stoga, (npr. frekvencija odašiljanja, protokoli za pristup mediju i sl.) kako bi se simuliralo ponašanje sustava što vjernije samoj implementaciji. Iako takav pristup, promatrajući iz perspektive same mreže, nema konceptualnih zamjerki, u slučaju kada je problem koji mreža treba riješti definiran generički (primjerice određivanje lokacija u neusidrenim mrežama u kojima čvorovi imaju mogućnost mjerenja udaljenosti) odabir tih parametara nije izravno vezan uz sam problem koji se rješava. Drugim riječima njihov izbor je proizvoljan što može dovesti do simulacije specifičnijeg scenarija nego što je potrebno.

U ovom poglavlju opisan je izvorni programski simulator raspodijeljenih algoritama u bežičnim mrežama osjetila koji funkcionira na aplikacijskoj razini opisan u radu autora [74]. Simulator je napisan u programskom jeziku Python u formi programske knjižnice. Python je objektno orijentiran skriptni jezik, jednostavne i minimalističke sintakse čije najveće adute čini ogroman broj kvalitetnih knjižnica za razne namjene, između ostalih i one za znanstveno računanje te posljedično sve veća popularnost unutar znanstveno istraživačke zajednice. Funkcije koje pruža simulator implementirane su bez dodatnog sloja apstrakcije čime se koristi prirođena snaga ekspresivne sintakse samog jezika. Korištenjem simulatora, korisnici mogu brzo i točno implementirati i simulirati raspodijeljene algoritme.

Simulator se posebice fokusira na sljedeće:

1. brza i jednostavna implementacija ideja i pristupa na algoritamskoj razini, bez po-

trebe za specifikacijama parametara nevezanih uz problem, korištenjem formalno definiranog raspodijeljenog računalnog okruženja.

2. podrška za dva različita procesa rada (engl. *workflow*) i za njihovu jednostavnu integraciju: (1) *interaktivni* način kojim se kontrolira izvršenje simulacije korak po korak i omogućava jednostavna inspekcija i modifikacija svih objekata za vrijeme izvršavanja, te (2) *potpuno automatizirano* stvaranje i izvršavanje simulacija u cilju provođenja eksperimenata na intuitivan način, korištenjem jednostavnih Python skripti.
3. promoviranje reproducibilnog istraživanja u skladu s principima otvorenog koda (engl. *open source*) te posljedično poticanje ponovnog iskorištenja i vrednovanja istog, kroz usporedbu s novim idejama, koristeći jednostavno prilagodljive kriterije.

S obzirom da simulator koristi formalno definirano raspodijeljeno računalno okruženje, implementacija pojedinog algoritma je jednostavan i izravan proces. Ovaj proces je dodatno olakšan korištenjem interaktivne konzole i ugrađenog programa za ispravljanje pogrešaka (engl. *debugger*) unutar kojeg se može izravno pristupiti svim objektima i izvršiti njihovo ispitivanje i eventualnu modifikaciju.

Kôd u Pythonu obično je organiziran unutar modula, pa je napredno korištenje, ili eventualno proširenje osnovnih funkcija, omogućeno pisanjem dodatnih modula i nasljeđivanjem od osnovnih razreda definiranih u Pymote knjižnici. Pisanje ovakvih proširenja je poticano kroz razvojni proces otvorenog koda.

U nastavku poglavlja kratko su spomenuti i analizirani trenutno dostupni simulatori. U dijelu 7.2 formalno se definira raspodijeljeno računalno okruženje te se daje teorijska podloga i osnovni principi na kojima se zasniva simulator. Nakon toga slijedi kratka diskusija s temom odabira platforme, opisom implementacijskih detalja i načina na koje se može obaviti nadogradnja simulatora novim mogućnostima. Naposlijetku, u dijelu 7.4 daje se primjer korištenja, odnosno opisuje se na koji se način može definirati, simulirati i analizirati jedan od popularnih algoritama za određivanje položaja koristeći interaktivnu konzolu i automatizirane eksperimente.

7.1 Pregled postojećih simulatora

U postojećoj literaturi opisan je veliki broj simulatora u kojima se mogu implementirati i analizirati algoritmi za bežične mreže osjetila. Ti simulatori fokusiraju se na različite ciljeve te se zbog toga prilično razlikuju po razini složenosti i uključenim funkcijama. Nadalje, oni podržavaju različite hardverske platforme, različite komunikacijske slojeve i njihove protokole, različite implementacije raspodijeljenih mreža i njhovih okruženja, te dolaze s različitim skupovima alata za modeliranje, analizu i vizualizaciju. U skupinu

klasičnih simulatora uključeni su NS-2, OMNeT++, J-Sim, TOSSIM i drugi.

NS-2 je simulator diskretnih događaja za generičko simuliranje svih tipova mreža i predstavlja standard u području. Simulator omogućava simulaciju protokola kao što su protokoli transportnog sloja (UDP, TCP i dr.), protokoli za usmjeravanje i protokoli sloja podatkovne veze kao što su protokoli za pristup mediju (engl. *media access protocol – MAC*) za klasične žične i bežične mreže. U svojoj osnovnoj inačici fokusiran je na mreže zasnovane na internet protokolu (IP), ali postoje i razna proširenja za bežične mreže osjetila [75]. NS-2 omogućava detaljno skupljanje podataka za vrijeme simulacije iz kojih se, korištenjem alata za animaciju (engl. *network animation – NAM*), pruža mogućnost za naknadnu vizualnu analizu. S obzirom na vrlo detaljnu simulaciju na razini paketa NS-2 nije primjeren za simulaciju velikih mreža koje se sastoje od više tisuća čvorova. Dodatni nedostatak je vrlo zahtjevna krivulja učenja i korištenja koja zahtjeva značajan napor kako bi se ispravno definiralo simulacijsko okruženje i proveli eksperimenti te analizirali rezultati. Jezgra simulatora i većina mrežnih protokola napisani su u programskom jeziku C++, dok se za definiciju i konfiguraciju simulacijskog okruženja koristi OTcl. NS-2 dolazi u velikom broju različitih inačica i distribucija te s različitim skupom proširenja, većinom uz licence otvorenog koda.

Jedno od proširenja simulatora je Mannasim radni okvir koji uvodi nove modele za projektiranje, razvoj i analizu različitih aplikacija za bežične mreže osjetila. Mannasim radni okvir pruža standardizirane strukture za definiciju različitih tipova čvorova kao što su: regularni čvor mreže, glava grozda i pristupna točka. Ta tri tipa čvora unutar simulacije izvršavaju različite algoritme koji su implementirani izravno u C++ programskom jeziku.

Sljedeća, u znanstvenoj zajednici vrlo popularna i korištena, platforma je OMNeT++ [76]. To je proširiva, modularna i komponentno orijentirana C++ knjižnica i radni okvir koji se koristi primarno za izradu mrežnih simulatora. OMNeT++ knjižnica pruža podršku za prikupljanje simulacijskih podataka, njihovu analizu, statistiku te grafičku prezentaciju. Specifične funkcije kao što je podrška za *ad-hoc* mreže i bežične mreže osjetila, razni protokoli, simulacije u stvarnom vremenu, emulacije mreža, korištenje alternativnih programskih jezika, integraciju baza podataka i ostale funkcije i modeli omogućeni su u dodatnim radnim okvirima i razvijaju se kao nezavisni projekti. Jedno proširenje koje se često koristi za simulaciju bežičnih mreža osjetila je Castalia [77]. Razvoj proširenja je motiviran željom da se omogući što vjerniji model radio kanala. Ostale funkcije ovog proširenja uključuju MAC protokol koji se može parametrizirati, modele fizičkih procesa i osjetila te pomak sistemskog sata koje mogu otkriti moguće probleme u ponašanju algoritma u realnim okruženjima. S druge strane, za razvoj algoritama na visokoj razini i analizu njihovog ponašanja, Castalia je vremenski vrlo zahtjevna, upravo zbog velikog

broja parametara čiji fokus je na nižim komunikacijskim slojevima i okolini.

J-Sim je simulator opće namjene, napisan u programskom jeziku Java koristeći paradigmu zasnovanu na komponentama. Komponente su dijelovi simulacijskog sustava koji imaju malu međusobnu spregu te se mogu projektirati, implementirati i ispitivati potpuno neovisno. Na osnovama takve arhitekture implementiran je generički mrežni model s komutacijom paketa u kojem je definirana struktura čvora i generičke mrežne komponente koje mogu biti korištene kao osnovni razredi za implementaciju protokola u različitim slojevima komunikacijskog stoga. J-Sim je inicijalno isprojektiran za simulaciju žičanih mreža, no postoje ekstenzije za bežične mreže koje implementiraju IEEE 802.11 MAC protokol koji u kombinaciji sa skupom ostalih protokola omogućava simulaciju bežičnih mreža.

TOSSIM [78] je knjižnica za simulacije mreža specifično namijenjena za simulacije mreža čiji čvorovi funkcioniraju koristeći TinyOS [79] operacijski sustav. Ova knjižnica omogućava emulaciju komponenti sustava i modeliranje različitih mrežnih topologija te na taj način pruža realistično okruženje za ispitivanje komunikacijskog opterećenja algoritma. Velika prednost korištenja ovog simulatora u sprezi s TinyOS-om je što se programski kod kojim se implementira algoritam na stvarnim čvorovima može izravno prevesti u izvršnu datoteku koja se može izvesti na standardnom osobnom računalu. Također TOSSIM podržava simulacije do nekoliko tisuća virtualnih čvorova, a dolazi u paketu s grafičkim korisničkim sučeljem TinyViz kroz koji se može vizualizirati simulacija i obavljati interakcija. TOSSIM je distribuiran pod licencom zasnovanom na principima otvorenog koda.

Neka od istraživanja u kojima se mogu pronaći informacije u vezi navedenih sustava i njihove usporedbe su [80, 81, 82]. Jedna od klasifikacija koja se može naći u literaturi [83] dijeli simulatore u tri glavne kategorije:

1. Algoritamska razina
2. Razina paketa
3. Razina instrukcije.

Simulatori algoritamske razine fokusiraju se na programsku logiku, podatkovne strukture i prezentaciju. Ovaj tip simulatora ne uzima u obzir detalje komunikacijskog modela, već se najčešće zasniva na strukturi grafa za prezentaciju komunikacijskih veza među čvorovima. Simulatori koji djeluju na razini paketa implementiraju sloj podatkovne veze i fizički sloj komunikacijskog stoga. Tako je uobičajeno za ovaj tip simulatora da sadrže implementacije 802.11b protokola i radio modele koji uključuju propagacijske efekte kao što su kratkotrajni propadi jakosti signala (engl. *fading*), interferencija, šum i ogib. Simulatori na razini instrukcije dodatno modeliraju rad procesora na razini instrukcije ili ciklusa, a često se nazivaju i emulatori.

Prema ovoj klasifikaciji, Pymote je simulator algoritamske razine, odnosno ne pruža simulaciju najnižih komunikacijskih slojeva niti rada procesora. Umjesto toga, Pymote koristi apstrakni model komunikacijskih entiteta i okoline te na taj način omogućava korisniku fokusiranje na općenite principe koji nisu pod utjecajem implementacijskih detalja. Ova činjenica Pymote čini jednostavnijim za učenje i korištenje. Dodatna korist ovakve implementacijske odluke je mogućnost simuliranja mreža s velikim brojem čvorova. Fokus simulatora je na projektiranje i vrednovanje algoritama te pružanje alata potrebnih za jednostavnu i brzu definiciju različitih mrežnih struktura. U nastavku su spomenuti i neki od postojećih simulatora algoritamske razine koji se koriste u znanstveno istraživačkoj zajednici.

AlgoSensim je radni okvir koji se koristi za simulaciju raspodijeljenih algoritama s fokusom na specifične zadaće poput određivanja položaja, usmjeravanja i distribucije podataka. Napisan je u programskom jeziku Java, a za konfiguraciju koriste se XML dokumenti. Objavljen je 2006 godine pod licencom otvorenog koda no od tada ostaje u prvotnoj ranoj radnoj verziji.

Shawn [84] je simulator napisan u programskom jeziku C++ čiji primarni cilj je simulirati efekt koji uzrokuje neka pojava, a ne simulirati samu pojavu, te na taj način povećati skalabilnost i dati podršku za slobodan izbor modela implementacije. Primjerice, umjesto da se provede cjelokupna simulacija protokola za pristup mediju uključujući modeliranje i simulaciju propagacije radio valova, Shawn simulira efekte kao što su gubitak paketa, korupcija podataka unutar paketa i kašnjenje. Na ovaj način se omogućava efikasnija implementacija dok je efekt na aplikacijski sloj sličan onome koji bi bio uz simulaciju propagacijskih efekata kao što su gušenje, višestazno širenje i drugi. To u konačnici omogućava simulaciju mreža s velikim brojem čvorova.

NetTopo [83] je integrirani radni okvir za simulaciju i vizualizaciju bežičnih mreža osjetila napisan u Javi. Glavni cilj i fokus ovog radnog okvira je simulacija aplikacija koji se mogu izvršavati jednim djelom u simulacijskom okruženju, a drugim u čvorovima bežične mreže. Omogućava simulaciju mreža s velikim brojem čvorova te implementira grafičko sučelje za pokretanje i kontrolu simulacija.

Sinalgo je još jedan radni okvir napisan u Javi koji djeluje na aplikacijskoj razini. S obzirom da ne simulira niže slojeve komunikacijskog stoga, zasniva se na komunikacijskom modelu definiranom na razini prosljeđivanja poruka, te na taj način također podržava simulaciju mreža s velikim brojem čvorova. Sinalgo dolazi u paketu sa skupom modela koji simuliraju mobilnost čvorova, njihovu povezanost i inicijalnu distribuciju koje korisnik može nadopuniti prema svojim potrebama. Simulacija obično započinje unutar grafičkog sučelja no isto tako se može koristiti skriptirani automatizirani način rada pogodan za veće eksperimente.

Bitna razlika između navedenih simulacijskih okruženja algoritamske razine i Pymote-a je programski jezik i okruženje. Pymote koristi prednosti programskog jezika Python kao što su jednostavnost korištenja i, veća ekspresivnost te posljedično brži razvoj u odnosu na rješenja u programskim jezicima C++ i Java, što čini to rješenje vrlo pogodnim za izradu prototipova. Dodatna prednost Pymote-a se očituje u tome što podržava i interaktivni i automatizirani način rada te na taj način omogućava brzo i često isprepleteno provođenje definicijske i simulacijske faze. Python nadalje omogućava jednostavan, a ujedno i vrlo napredan korisnički uvid u objekte u memoriji za vrijeme samog izvršavanja programa odnosno u ovom slučaju simulacije ili eksperimenta, te na taj način omogućava korisniku izravno eksperimentiranje i jednostavnije otkrivanje pogrešaka.

7.2 Raspodijeljeno računalno okruženje

Preduvjet za pisanje raspodjeljenog algoritma je egzaktna definicija okruženja u kojem taj algoritam treba riješiti zadani problem. Raspodijeljeno računalno okruženje i pretpostavljena ograničenja u domeni definicije problema kojeg treba riješiti, a koja su činila osnovnu specifikaciju Pymote simulatora uzeta su iz seminalne publikacije [62]. Principi kojima se regulira funkcioniranje algoritma u ovom okruženju opisani su nastavku.

Raspodijeljeno računalno okruženje sastoji se od skupa *računalnih entiteta* $\mathcal{E}$ koji u ovom slučaju predstavljaju čvorove bežične mreže te *poruka* koje ti entiteti izmjenjuju. Čvor $x \in \mathcal{E}$ ima mogućnost spremanja podataka u svoju ograničenu lokalnu memoriju M_x koja se sastoji od određenog broja imenovanih registara. Jedan od njih koji se naziva *statusni registar* ima specijalnu funkciju koja određuje ponašanje samog čvora i može poprimiti samo vrijednosti iz definiranog niza statusa $\mathcal{S}$. Ostali dijelovi čvora x su procesor i komunikacijski modul. Jedan od ciljeva efikasnog raspodijeljenog algoritma je njihovo što manje korištenje.

Ponašanje čvora unutar ovog računalnog okruženja je u potpunosti reaktivno, odnosno čvor djeluje samo ukoliko detektira pojavu jednog od tri moguća događaja: (1) primitak poruke, (2) istjecanje prethodno postavljenog alarma i (3) spontani impuls. Spontani impuls je definiran kao bilo koji događaj čiji uzrok je izvan same mreže, a koji je čvor sposoban detektirati. Jedan primjer spontanog impulsa je korisničko pritiskanje gumba na samom čvoru. Spontani impuls se često koristi za inicijalizaciju algoritama.

Odabir akcije koju čvor treba obaviti je izravni rezultat njenog statusa (vrijednosti zapisane u statusnom registru) i tipa događaja što prikazuje sljedeći izraz:

$$\text{status} \times \text{događaj} \longrightarrow \text{akcija} \tag{7.1}$$

Tablica 7.1: Struktura poruke.

Ime polja	**Tip podataka**	**Opis**
source	čvor	čvor pošiljatelj
destination	čvor	čvor kome je namijenjena poruka
nexthop	čvor	opcionalno instanca čvora koja je sljedeća na putu prema odredištu
header	niz znakova	zaglavlje poruke koje definira strukturu odaslanih podataka u polju data i ujedno samom algoritmu služi za odabir akcije koju će čvor izvršiti po primitku poruke
data	mapa	bilo koji podaci čija struktura je definirana s obzirom na zaglavlje poruke u samom algoritmu

Raspodijeljeni algoritam je definiran kao skup pravila koji pridružuju sve moguće kombinacije statusa i tipova događaja specifičnim akcijama. Za raspodijeljeni sustav se kaže kako ima homogeno ponašanje ukoliko svi entiteti tog sustava obavljaju isti algoritam što posljedično pojednostavljuje razvoj algoritma i njegovu analizu. Kako se svako nehomogeno ponašanje može učiniti homogenim, kao što je opisano u [62], svi čvorovi unutar Pymote simulatora po definiciji obavljaju isti algoritam što omogućava definiciju algoritma na razini mreže, a ne na razini čvora.

Čvor može obaviti tri različita tipa akcija: (1) primitak, spremanje i obrada podataka, (2) slanje podataka čvorovima u dometu u formi poruka i (3) promjena vrijednosti statusnog registra. Sve ostale akcije, kao što je primjerice mjerenje korištenjem priključenih osjetila može se smatrati kombinacijom primitka poruka te spremanja i obrade podataka.

Odašiljanje poruka između čvorova definira se kao prijenos ograničenog niza bitova. Jednostavna, a ujedno i generička struktura poruke koja se koristi unutar Pymote simulatora, te opis pojedinih polja unutar te strukture dana je u tablici 7.1.

Svaki čvor x može poslati poruku samo čvorovima susjedima koji su u dometu njegovog radio signala $N_{out}(x) \in \mathcal{E}$ i primiti poruku samo od čvorova u čijem je dometu $N_{in}(x) \in \mathcal{E}$. Pretpostavljena značajka svakog čvora je takozvana *lokalna orijentacija* što znači da svaki čvor može razlikovati svoje susjede prema njihovom jedinstvenom identifikacijskom broju (ID). Na taj način primjerice čvor može poslati poruku određenom čvoru susjedu bez da ju šalje ostalim susjedima što je u simulatoru omogućeno kroz polje `destination` u strukturi poruke.

U praksi, slanje i primanje poruka je složena operacija podložna pogreškama koja se

proteže kroz više slojeva komunikacijskog stoga s nepoznatim, a moguće i neograničenim, trajanjem. Prilikom projektiranja raspodijeljenih algoritama vrlo je važno učiniti njihovo funkcioniranje neovisnim od pogrešaka u komunikaciji i neovisnim od komunikacijskog kašnjenja.

U dosadašnjem opisu okruženja sva svojstva mreže bila su definirana za generički slučaj. Posebna svojstva okruženja, odnosno uvjete u kojima funkcionira algoritam, nazivaju se *ograničenja* zato što su algoritmi koji su projektirani s pretpostavkom zadovoljenja tih uvjeta zapravo ograničeni u primjeni samo na slučajeve kada su ti uvjeti zadovoljeni, odnosno ne mogu biti primjenjeni u ostalim slučajevima.

Trenutna ograničenja koja su definirana u simulatoru su:

Dvosmjerne veze: za svaki čvor x, skup čvorova u dometu i skup čvorova kojima je čvor x u dometu je identičan $N_{out}(x) = N_{in}(x), \forall\ x\ \in\ \mathcal{E}$ i čvorovi koji pripadaju tom skupu se nazivaju *susjedima* od x.

Povezanost: topologija mreže je takva da svaki čvor može komunicirati s bilo kojim drugim čvorom u mreži, bilo izravno ili posredstvom drugih čvorova.

Potpuna komunikacijska pouzdanost: svaka poslana poruka će biti primljena u konačnom vremenu, te će njen sadržaj biti identičan, odnosno bez pogrešaka.

Ova ograničenja su prilično jaka, ali i razumna. Njihovom pretpostavkom se pokušava reducirati domena problema isključivo na aplikacijski sloj i na taj način odstraniti neželjene nepotrebne specifikacije, stavljajući fokus na generička rješenja postavljenog problema.

Sadržaj memorije čvora i informacija sadržana u toj memoriji, predstavlja takozvano *lokalno znanje* čvora. Ako barem jedan čvor u skupu čvorova $W \subseteq \mathcal{E}$ posjeduje informaciju p, onda je ta informacija po definiciji dio *implicitnog znanja* tog skupa. Ako svaki čvor u skupu W posjeduje informaciju onda je ona dio *eksplicitnog znanja* tog skupa. Poimanje znanja unutar mreže, odnosno dijela mreže ili pojedinog čvora, je vrlo bitno jer se cjelokupni smisao raspodijeljenih algoritama može definirati kao proces u kojem se povećava implicitno i eksplicitno znanje mreže.

7.3 Implementacija

Pymote simulator zasniva se na ciljevima opisanim na početku poglavlja i formalizmu raspodijeljenog računalnog okruženja. U nastavku teksta se razjašnjava odabir platforme, daje kratak opis implementacije onovnih funkcija simulatora, te se zaključuje s opisom mogućih proširenja.

7.3.1 Odabir platforme

Programski jezik Python s uključenim knjižnicama i alatima je odabran nakon analize postojećih platformi i radnih okvira. Nakon dugotrajnog i naprednog korištenja zaključak je kako Python u potpunosti zadovoljava sljedeće postavljene zahtjeve:

- Jednostavno i brzo učenje te dobra dokumentacija.
- Programski jezik s potpunim skupom mogućnosti koje pruža objektno orijentirana paradigma za razliku od ograničenih rješenja kao što je primjerice MATLAB.
- Jednostavan i ekspresivan jezik koji omogućava minimalistički programski kod u kojem je lako otkriti pogreške.
- Podrška za interaktivni način rada s obzirom da je taj način rada posebno pogodan za *ad-hoc* eksperimentiranje i analizu. **IPython** [85] je interaktivna konzola za Python koja pruža sve glavne funkcije potrebne za interaktivno znanstveno računanje (slika 7.1).
- Mogućnost introspekcije – programska inspekcija svih objekata i njihovih atributa.
- Bogata knjižnica za znanstveno računanje. Iako je Python generički programski jezik vrlo je popularan i konkurentan u tom području s obzirom na podršku istraživačke zajednice i pakete koje ta zajednica razvija i održava kao što su **NumPy** i **SciPy** [86] koji predstavljaju osnovne pakete za znanstveno računanje u Pythonu. Oni pružaju mnogobrojne mogućnosti iskoristive u interaktivnoj Python sjednici eksponirajući klase i metode za manipulaciju podacima na visokoj razini apstrakcije. Kao komplement ovim paketima, za grafičku prezentaciju rezultata analize koristi se paket **matplotlib**.
- Promocija reproducibilnog istraživanja na principima otvorenog koda: Python je platforma koja već promovira vrlo slične ideje i koja sa zajednicom korisnika i razvojnih programera napreduje u tom smjeru.

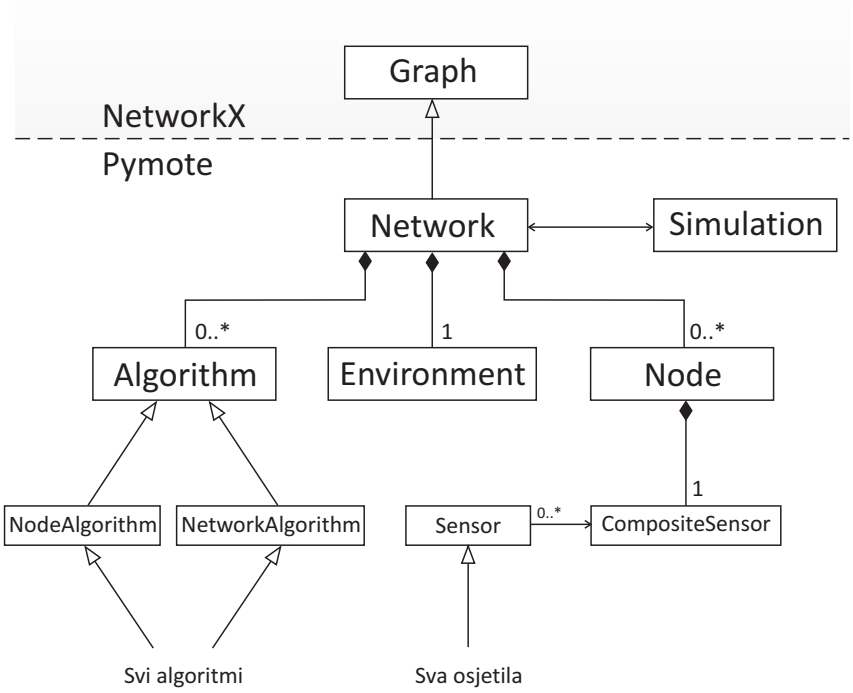
```
Python 2.7.1 (r271:86832, Nov 27 2010, 18:30:46) [MSC v.1500 32 bit (Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 0.13.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

Welcome to pylab, a matplotlib-based Python environment [backend: Qt4Agg].
For more information, type 'help(pylab)'.

In [1]:
```

Slika 7.1: IPython interaktivna konzola.

7.3.2 Osnovni razredi

Implementacijska filozofija slijedi principe DRY[1] i KISS[2]. Kako bi ostali vjerni DRY principu, Pymote knjižnica je pozicionirana tako da se iskoristi postojeći kod pri čemu se prvenstveno misli na osnovni razred simulatora `Network` koji s obzirom na odabrano raspodjeljeno računalno okruženje koje modelira mrežu kao graf nasljeđuje od razreda `Graph` iz paketa **NetworkX** [87].

NetworkX je paket za stvaranje, manipulaciju i proučavanje strukture, dinamike i funkcija kompleksnih mreža. Njegova definicija grafa je vrlo generička te omogućava jednostavno proširenje i nadogradnju. Dodatno, paket implementira veliki broj metoda i funkcija iz teorije grafova što ga čini idealnim kandidatom za proširenje. Nekolicina osnovnih razreda opisana je u nastavku. Pojednostavljeni dijagram razreda koji opisuje odnose među osnovnim razredima predstavljen je na slici 7.2.



Slika 7.2: Pojednostavljeni dijagram razreda.

Razred **Network** nasljeđuje od razreda Graph iz paketa Networkx i kao takav implementira sve metode definirane u tom razredu. Dodatno, svi algoritmi iz teorije grafova koji su implementirani unutar paketa kompatibilni su i s razredom `Network` i time izravno dostupni i u Pymote simulatoru. Dodatne funkcije razreda `Network` prvenstveno omogućavaju automatsko stvaranje komunikacijskih veza (bridova grafa) s obzirom na

[1]Ne ponavljaj se (engl. *do not repeat yourself – DRY*) je princip pisanja programskog kôda koji definira intenciju da Pymote programski kod ne reimplementira postojeće funkcionalnosti već da se razvija kao ekstenzija postojećih funkcija unutar ostalih Python paketa.

[2]Jednostavnost prije svega (engl. *keep it simple stupid – KISS*) je princip koji odražava potrebu da se pojednostavni korištenje simulatora i implementacija i integracija pripadajuće knjižnice.

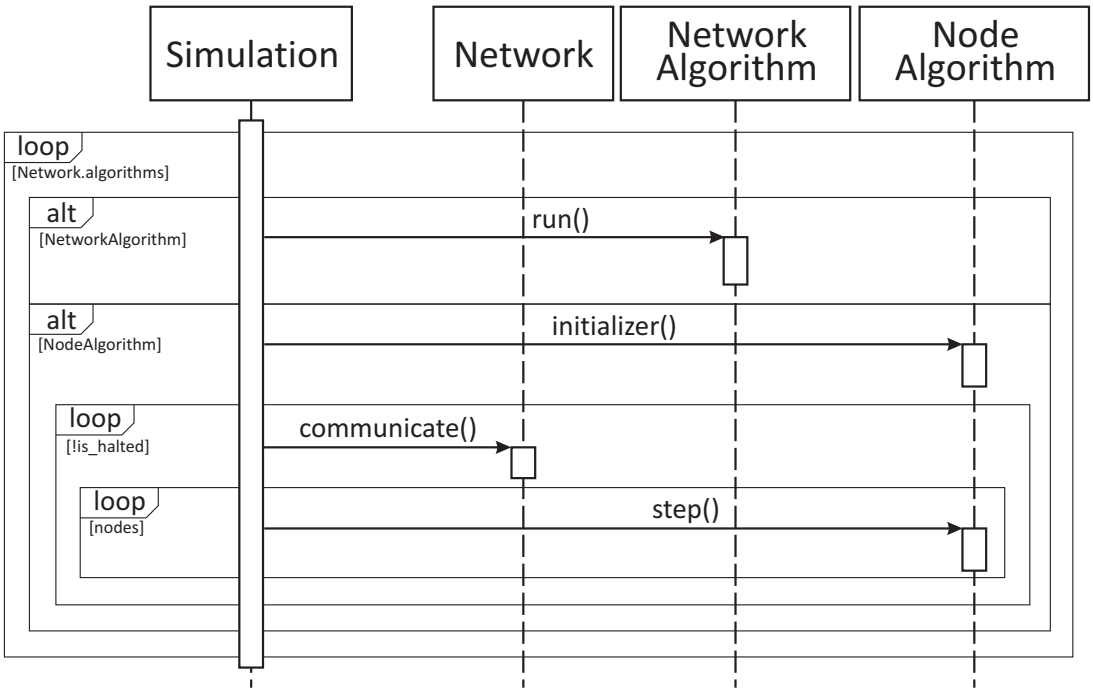
lokacije čvorova, okruženje i tip komunikacijskog kanala. Kao što je već spomenuto, s obzirom na homogeno ponašanje čvorova, upravo se na razini mreže odnosno razreda `Network` omogućava definiranje uređene liste instanci razreda `Algorithm` koja je opisana u nastavku. Ostale funkcije razreda `Network` uključuju implementaciju komunikacije odnosno prosljeđivanja poruka između čvorova susjeda i brigu za trenutno stanje algoritama koji se izvršavaju u čvorovima mreže.

Razred **Node** predstavlja bežični čvor mreže. Svaki čvor identificiran je svojim jedinstvenim brojem kako bi se zadovoljilo spomenuto svojstvo lokalne orijentacije. Dodatno, čvor sadrži skup polja u memoriji kao što su polje za odlazne poruke `outbox`, za primljene poruke `inbox`, statusno polje, te memorija opće namjene. Kako čvor posjeduje mogućnost komunikacije s čvorovima u dometu, atribut `commRange` definira komunikacijski domet koji koristi mreža kako bi odredila s kojim čvorovima je moguće ostvariti komunikaciju. Kako bi se znanje koje posjeduje čvor simuliralo čim vjernije, čvor ne zna ništa što nebi znao u stvarnoj implementaciji, primjerice čvor ne zna svoju lokaciju unutar okruženja. Ako bi kojim slučajem ta informacija trebala biti dijelom znanja pojedinog čvora, npr. ako je čvor sidro, tada korisnik može u postavkama eksperimenta opremiti dotični čvor s adekvatnim osjetilom koji će mu pružiti tu informaciju s određenom razinom točnosti koju je moguće definirati u postavkama osjetila, kao što je demonstrirano u dijelu 7.4.2.

Instance razreda **Simulation** se koriste za kontrolu izvršavanja algoritama definiranih na razini mreže, korak po korak. Simulacija koristi metode definirane u razredu `Algorithm` kako bi ih izvršavala ili na razini čvora ili na razini mreže kao što je prikazano na sekvencijskom dijagramu na slici 7.3. Za vrijeme izvršavanja algoritma, stanje mreža i pripadajućih čvorova se mijenja, ali sama instanca simulacije ne zapisuje niti sadrži podatke o trenutnom stanju. Svi podaci prije, za vrijeme i nakon simulacije sadržani su unutar intance mreže što je bitno s obzirom da se tako u bilo kojem trenutku simulacije svi relevantni podaci mogu jednostavno spremiti u datoteci serijalizacijom instance mreže.

Razred **Algorithm** predstavlja kôd koji se izvršava za vrijeme simulacije unutar mreže. Trenutno postoje dva glavna podrazreda ovog razreda, a to su `NetworkAlgorithm` i `NodeAlgorithm`.

Razred **NodeAlgorithm** implementira klasični raspodijeljeni algoritam koji se izvršava u svakom čvoru mreže. Početak izvršavanja izazvan je spontanim impulsom u slučajno ili specifično odabranim čvorovima, a koji se definira u metodi `initialize`. Svaka akcija koja slijedi nakon toga predstavlja rezultat nadolazeće poruke i statusa u kojem se čvor nalazi (istjecanje alarma još uvijek nije implementirano) kako je definirano u raspodijeljenom računalnom okruženju.



Slika 7.3: Sekvencijski dijagram izvršenja algoritma.

Postoji vrlo bitna razlika između lokalnog završetka algoritma pri čemu čvor može odrediti trenutak u kojem je gotov s izvršavanjem svih akcija u trenutnom algoritmu i globalnog završetka u kojem čvor zna da su i svi ostali čvorovi također završili s izvršavanjem trenutnog algoritma. Samo globalni završetak može predstavljati nedvosmislen signal koji će rezultirati početkom izvršavanja sljedećeg algoritma u nizu. U Pymote simulatoru instanca simulacije koja se koristi za izvršenje algoritama u mreži je odgovorna i za detekciju završetka i to detekcijom nepostojanja poruka (lokalni završetak) ili eksplicitnim signalom dobivenim od strane jednog čvora (globalni završetak).

Kako bi se pojednostavnila definicija i izvršavanje raspodijeljenih algoritama koji se koriste samo kao pomoćni algoritmi za unaprijeđenje znanja čvorova, a u cilju zadovoljenja preduvjeta algoritma koji se ispituje, unutar simulatora se omogućava implementacija njihove centralizirane verzije u obliku instance razreda `NetworkAlgorithm`. Takvim algoritmima omogućava se unaprijeđenje znanja svih čvorova izravnim unosom podataka u njihovu memoriju. Primjerice, centralizirana verzija raspodijeljenog algoritma za stvaranje razapinjajućeg stabla ima mogućnost zapisivanja odnosa dijete roditelj izravno u memoriju svih čvorova mreže. Algoritmi koji se ispituju bi uvijek trebali biti implementirani kao klasični raspodjeljeni algoritmi odnosno kao podrazredi razreda `NodeAlgorithm`.

7.3.3 Proširivanje osnovnih funkcija

Kao što je već napomenuto u uvodnom dijelu, Pymote je primarno fokusiran na simulaciju algoritama na aplikacijskoj razini. Drugim riječima, detalji komunikacije među

entitetima ne utječu na rezultat algoritma što je osigurano uvođenjem ograničenja kao što su potpuna pouzdanost, opisana u dijelu 7.2.

Iako ne utječu na rezultat, niži slojevi komunikacijskog stoga utječu na performanse mreže kao što je primjerice iskorištenje ograničenih izvora energije. Ukoliko korisnik želi simulirati i analizirati utjecaj specifičnih protokola nižih slojeva to je u svakom slučaju moguće učiniti. Primjerice simulacija i naknadna analiza potrošnje energije može biti obavljena nasljeđivanjem razreda `Network` i nadjačavanjem (engl. *override*) metode `communicate`.

Dodatni primjer je simulacija radio kanala koja može biti obavljena pisanjem vlastitog razreda **ChannelType** koji se zatim proslijedi instanci mreže. Knjižnica simulatora trenutno implementira dva jednostavna podrazreda `ChannelType` razreda: **Udg** – graf jediničnih kružnica (engl. *unit disc graph – UDG*) u kojem su čvorovi u vezi kada su jedan drugome unutar komunikacijskog dometa te **SquareDisc** u kojem je vjerojatnost veze između dva čvora jednaka $1 - d^2/r^2$, gdje je d udaljenost između čvorova, a r je njihov komunikacijski domet.

Najjednostavniji primjer proširenja je protokol za usmjeravanje koji se može implementirati kao raspodjeljeni algoritam s ciljem održavanja tablice usmjeravanja u čvoru memorije. Podaci u tablici mogu biti iskorišteni za adekvatno popunjavanje polja `nexthop` koje je prisutno u strukturi poruke (tablica 7.1) i koje se koristi u metodi `communicate` prilikom slanja poruke susjednom čvoru.

Svi ovi primjeri proširenja mogu se integrirati u budućim verzijama simulacijske knjižnice što se posebice potiče korištenjem kolaboracijskog razvojnog procesa zasnovanog na principima otvorenog koda.

7.4 Primjer korištenja: Implementacija algoritma

Kao što je već spomenuto Pymote podržava dva načina rada. Jedan od najpopularnijih načina rada u istraživačkoj zajednici je korištenje interakativne konzole. Ovaj način rada omogućava brzo i efikasno eksperimentiranje, jednostavnu inspekciju te izravan pristup i modifikaciju svih objekata. Drugi popularan način rada je izvršavanje serije pripremljenih i automatiziranih eksperimenata. U ovom dijelu predstavljeni su primjeri korištenja simulatora na oba navedena načina kroz implementaciju već spomenutog (poglavlje 3.2.4) i opisanog algoritma za određivanje lokacija DV-hop [88].

7.4.1 Definicija algoritma

Unutar Pymote simulatora algoritmi se definiraju kao izravne Python implementacije. Za primjer je uzeta jednostavna ideja algoritma DV-hop: (a) svaki čvor u mreži treba

estimirati svoju udaljenost sa čim više čvorova sidara i (b) uz poznate estimirane udaljenosti i lokacije sidara, čvor može estimirati svoju lokaciju korištenjem trilateracije.

Algoritam se sastoji od dvije faze:

1. Sva sidra preplavljuju (engl. *flooding*) mrežu informacijom o svojoj lokaciji. Tijekom propagacije poruka kroz mrežu svaki čvor povećava brojilo komunikacijskih skokova za svaku poruku i održava trenutno zabilježeni minimum za svako sidro posebno. Za vrijeme ove faze, sidra računaju prosječnu duljinu komunikacijskog skoka tako što dijele poznatu euklidsku udaljenost od ostalih sidara minimalnim zabilježenim brojem skokova za pojedino sidro.
2. Nakon što je prva faza uspješno završila, svako sidro obavlja još jedno kontrolirano preplavljivanje informacijom o novoj, estimiranoj duljini komunikacijskog skoka. Nakon prvog primitka estimirane duljine skoka od najbližeg sidra pojedini čvor ga prosljeđuje susjedima i ignorira sve daljnje primljene poruke. Nakon toga estimira udaljenosti od sidara i svoju lokaciju.

Ove dvije faze implementirane su kao dva odvojena algoritma odnosno dvije instance razreda koji nasljeđuje od razreda `NodeAlgorithm`. Kako obje faze koriste isti protokol za preplavljivanje zgodno je definirati jedan generički algoritam za tu namjenu te za svaku od faza nasljediti od njega, te nadjačati odgovarajuće metode. U ispisu 7.1 navedena je implementacija osnovnog protokola za preplavljivanje.

Općenito, `NodeAlgorithm` je strukturiran kao skup funkcija koje odgovaraju svim statusima koje čvor može poprimiti za vrijeme izvršavanja pojedinog algoritma. Svaka od funkcija treba omogućiti obradu svih tipova događaja što trenutno znači obradu dolaznih poruka (definiranih zaglavljem) za koje je potrebno obaviti određenu akciju.

Kako je ovo jednostavan protokol u kojem čvorovi ostaju uvijek u jednom statusu FLOODING i kako postoji samo jedan tip poruke – onaj sa zaglavljem "Flood", implementacija je vrlo jednostavna.

Treba primjetiti kako se na tri mjesta unutar samog algoritma obavljaju pozivi funkcija `initiator_condition`, `initiator_data` i `handle_flood_message`. To su funkcije koje algoritmi koji nasljeđuju od ovog osnovnog algoritma mogu nadjačati i redefinirati kako bi obavili specifičnu namjenu u danom trenutku što se upravo i čini unutar algoritama koji implementiraju prvu odnosno drugu fazu DV-hop algoritma. Ova redefinicija predstavljena je u ispisima 7.2 za prvu fazu i 7.3 za drugu fazu.

Ovakva definicija je u skladu sa spomenutim DRY principom, štoviše, svi ostali algoritmi koji žele koristiti preplavljivanje mogu iskoristiti razred FloodingUpdate jednostavnim nasljeđivanjem, nadjačavanjem i svojom vlastitom implementacijom spomenutih funkcija. Kako vidimo, objektno orijentirana paradigma koja je prirođena Pythonu može se iskoristiti čak i u implementaciji raspodijeljenih algoritama unutar samog simulatora.

```
class FloodingUpdate(NodeAlgorithm):
  required_params = ('dataKey',)
  default_params = {}

  def initializer(self):
    for node in self.network.nodes():
      if self.initiator_condition(node):
        msg = Message(destination=node,header='initialize')
        self.network.outbox.insert(0,msg)
      node.status = 'FLOODING'

  def flooding(self, node, message):
    if message.header=='initialize':
      node.send(Message(header='Flood',
                        data=self.initiator_data(node)))

    if message.header=='Flood':
      updated_data = self.handle_flood_message(node,message)
      if updated_data:
        node.send(Message(header='Flood',
                          data=updated_data))

  STATUS = {'FLOODING' : flooding,}
```

Ispis 7.1: Generički protokol za preplavljivanje. Funkcija `initializer` je specifična funkcija koja simulira spontani impuls u obliku specijalne *inicijalizacijske* poruke na početku izvršavanja algoritma.

```
class DVHop(FloodingUpdate):

  def initiator_condition(self, node):
    node.memory[self.truePositionKey] = node.compositeSensor.read(node).get('TruePos',None)
    return node.memory[self.truePositionKey] is not None

  def initiator_data(self, node):
    return {node: concatenate((node.memory[self.truePositionKey][:2],[1]))}

  def handle_flood_message(self, node, message):
    if not node.memory.has_key(self.dataKey):
        node.memory[self.dataKey] = {}
    updated_data = {}
    for landmark,landmark_data in message.data.items():
      if landmark==node: continue
      # osvjezi samo ako je podatak novi ili je broj skokova manji
      if not node.memory[self.dataKey].has_key(landmark) or landmark_data[2]<node.memory[self.dataKey
          ][landmark][2]:
        node.memory[self.dataKey][landmark] = array(landmark_data)
        # povecaj brojac skokova
        landmark_data[2] += 1
        updated_data[landmark] = landmark_data
    # sidra trebaju rekalkulirati duljinu skoka
    if node.memory[self.truePositionKey] is not None:
      self.recalculate_hopsize(node)
   return updated_data
```

Ispis 7.2: DV-hop 1. faza: DVHop. Funkcija `recalculate_hopsize` je izostavljena zbog sažetosti ispisa. Potrebno je primjetiti kako tijekom izvršavanja funkcije `initiator_condition` čvor očitava TruePosSensor ukoliko je čvor opremljen s istim.

```
class Trilaterate(FloodingUpdate):

  def initiator_condition(self, node):
    return node.memory[self.truePositionKey] is not None

  def initiator_data(self, node):
    return node.memory[self.hopsizeKey]

  def handle_flood_message(self, node, message):
    if node.memory.has_key(self.hopsizeKey):
      return None
    node.memory[self.hopsizeKey] = message.data
    self.estimate_position(node)
    return node.memory[self.hopsizeKey]
```

Ispis 7.3: DV-hop 2. faza: trilateracija. Funkcija `estimate_position` je izostavljena zbog sažetosti ispisa.

7.4.2 Simulacija

Najbolji način za testiranje i, prema potrebi, otklanjanje pogrešaka u raspodijeljenim algritmima je korištenjem interaktivne konzole. U ispisu 7.4 predstavljen je jedan primjer takve interaktivne sjednice.

Dodatna prednost izvođenja simulacije unutar interaktivne konzole je što, ukoliko se dogodi pogreška, korisnik preuzima kontrolu. Naknadnim pokretanjem konzole za ispravljanje pogrešaka naredbom `%debug` dobija se mogućnost inspekcije programskog stoga kakav je bio u trenutku kada se pogreška dogodila i analize svih objekata u memoriji, pa čak i mijenjanja njihovih vrijednosti ili interaktivnog poziva funkcija i metoda.

Kao alternativa kontinuiranom izvršavanju, algoritmi mogu biti izvršavani i korak po korak. Ovaj način pokretanja i izvršavanja može biti vrlo koristan kada se želi dubinski analizirati ponašanje algoritma i obaviti inspekcija ili modifikacija objekata za vrijeme izvršavanja.

Osim toga, ovaj način može biti prikladan u situacijma kada se algoritam ne izvršava prema očekivanjima, ali pogreška nije dovoljno eksplicitna kako bi izazvala izlazak iz programa uslijed pogreške. Primjer ovog načina izvršavanja dan je u ispisu 7.5.

Drugi način rada u simulatoru je provođenje automatiziranog eksperimenta. Automatizirani eksperiment je definiran kao jednostavna Python skripta koja generira potrebne mreže, izvodi simulacije i naposlijetku ih sprema u datoteke za kasniju analizu. Primjer jedne takve skripte dan je u ispisu 7.6.

7.4.3 Analiza dobivenih podataka

Objekti unutar simulacije i dobiveni podaci mogu biti analizirani u bilo kojem trenutku. Primjerice, za vrijeme faze u kojoj se postavlja mreža ili nakon što je mreža učitana iz spremljene datoteke, prikladno je vizualizirati topologiju mreže. Iz konzole to je moguće obaviti pozivom metode `show` koja je definirana za instance razreda `Network`. Rezultat

```
# generiranje mreze sa 100 do 300 slucajno postavljenih cvorova i stupnjem 9
In [1]: netgen = NetworkGenerator(n_min=100,n_max=300,degree=9)
In [2]: net = netgen.generate()
# odabir sidara tako sto se odabranim cvorovima postavlja TruePosSensor
In [3]: for node in net.nodes()[:10]:
  ....:   node.compositeSensor = CompositeSensor(('TruePosSensor'))
# importiranje algoritama postavljanje u mrezu zajedno s parametrima
In [4]: from pymote.algorithms.niculescu2003.dvhop import DVHop
In [5]: from pymote.algorithms.niculescu2003.trilaterate import Trilaterate
In [6]: net.algorithms = \
  ....:    ((DVHop,
  ....:         {'dataKey': 'dvData',
  ....:          'truePositionKey':'landmarkPos',
  ....:          'hopsizeKey':'hopsize',
  ....:         }),
  ....:     (Trilaterate,
  ....:        {'dataKey': 'dvData',
  ....:         'positionKey':'dvHop',
  ....:         'truePositionKey':'landmarkPos',
  ....:         'hopsizeKey':'hopsize',
  ....:         }),
  ....:    )
In [7]: sim = Simulation(net)
In [8]: sim.run()
INFO     [simulation.py]: Simulation has finished.
# spremanje mreze i svih relevantnih podataka u datoteku za kasniju analizu
In [9]: write_pickle(net,'net.gz')
```

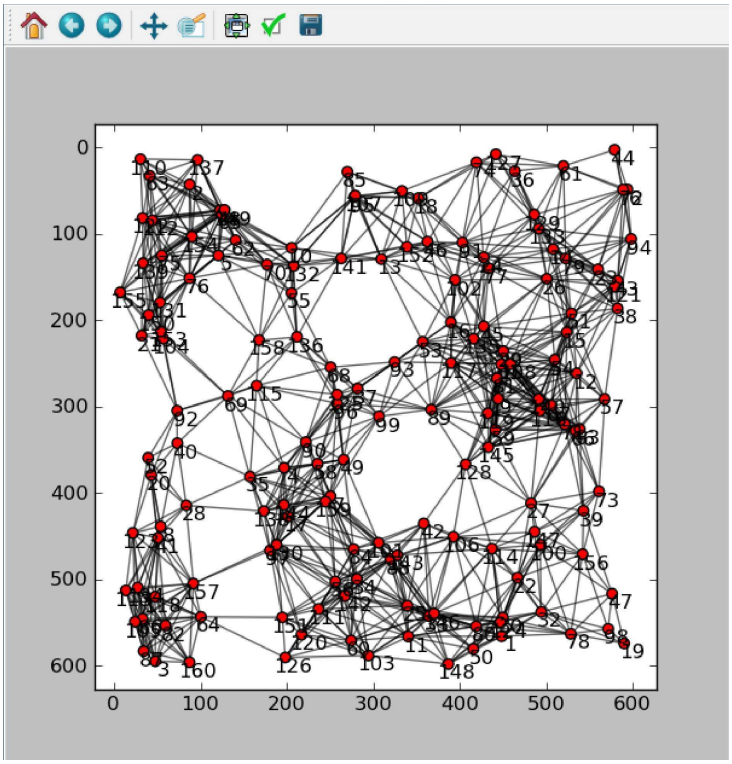
Ispis 7.4: Interaktivna sjednica za preliminarno simuliranje algoritma. Na početku se postavlja slučajno generirana mreža u standardnom okruženju. Izvorni članak u kojem je predstavljen algoritam koristi mrežu s 200 slučajno rapoređenih čvorova i prosječnim stupnjem povezanosti (brojem susjeda) 9, stoga se instancira razred **NetworkGenerator** s danim postavkama te se nakon instanciranja koristi za generiranje adekvatne mreže. Nakon toga, instancira se prethodno implementiran algoritam i postavlja u listu algoritama mreže. Kako algoritmi zahtjevaju postojanje čvorova sidara, prvih 10 čvorova u mreži se oprema s odgovarajućim osjetilom. Mreža je sad spremna za simulaciju, pa nakon instanciranja razreda Simulation može se pokrenuti i sama simulacija. Naposlijetku se u datoteku serijalizira i sprema instanca mreže na kojoj je obavljena simulacija kako bi se naknadno mogla obaviti analiza podataka spremljenih u memoriji čvorova kao što su u ovom primjeru estimirane lokacije.

```
In [10]: sim.reset()  # najprije je potrebno resetirati mrezu odnosno stanje algoritma i obrisati
     memoriju cvorova
In [11]: sim.run(1)  # izvrsi 1. korak prvog algoritma
In [12]: landmark_node = net.nodes()[0]
In [13]: landmark_node.inbox  # sidro prima inicijalizacijsku poruku
Out[13]:
[
------ Message ------
     source = None
destination = <Node id=1>
     header = 'initialize'
id(message) = 0x908e9f0>]
In [14]: sim.run(1)  # izvrsi slijedeci korak
In [15]: landmark_node.outbox  # sidro priprema poruku za slanje svim susjedima
Out[15]:
[
------ Message ------
     source = <Node id=1>
destination = Broadcasted
     header = 'Flood'
id(message) = 0x90a3d30>]
In [16]: landmark_node.outbox[0].data
Out[16]: {<Node id=1>: array([ 201.2419,  141.9482,    1.    ])}
# provjera ako je poslana lokacija ista kao ona koju je ocitao TruePosSensor
In [17]: landmark_node.memory['landmarkPos']
Out[17]: array([ 201.2419,  141.9482])
# jos jedna provjera izravnim ispitivanjem atributa mreze
In [18]: net.pos[node]
Out[18]: array([ 201.2419,  141.9482])
```

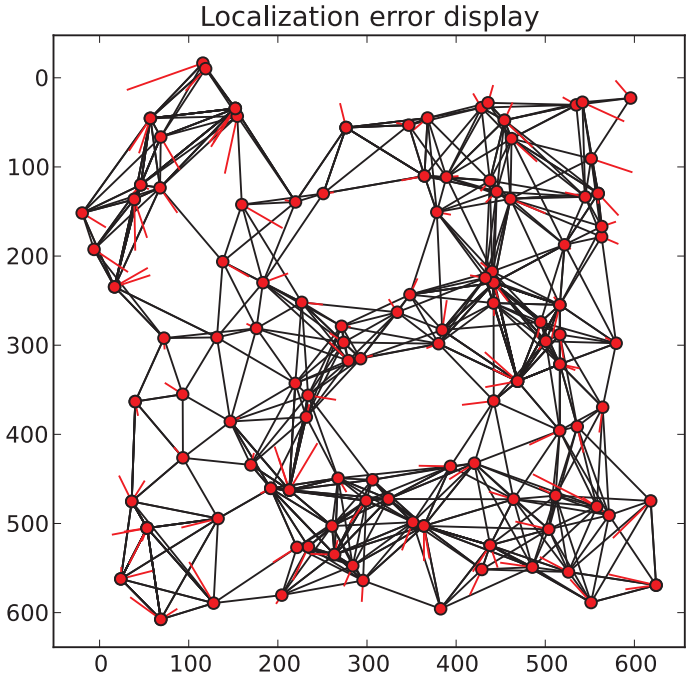
Ispis 7.5: Primjer izvršavanja simulacije odnosno odgovarajućih algoritama korak po korak te inspekcija čvorova i podataka zapisanih na razini mreže. U prvom koraku izvršavanja algoritma DV-hop spontani impuls je izazvan u čvorovima sidrima u obliku specijalnih inicijalizacijskih poruka. Kao posljedica, u drugom koraku sidra odašilju svim čvorovima susjedima *Flood* poruku u kojoj su zapisani podaci o lokaciji sidra i postavljenim brojilom broja komunikacijskih skokova na 1. Nakon toga provjerava se ako su to isti podaci koji su očitani od strane TruePositionSensor osjetila i na kraju inspekcijom odgovarajućeg atributa mreže provjerava se ako je to doista lokacija sidra.

```
netgen = NetworkGenerator(degree=9, n_min=100, n_max=300)
for lm_pct in [5,10,20,33]:
  for net_count in range(100):
    net = netgen.generate()
    for node in net.nodes()[:int(lm_pct*len(net.nodes())/100)]:
      node.compositeSensor = CompositeSensor(('TruePosSensor'))
    net.algorithms = ALGORITHMS
    sim = Simulation(net)
    sim.run()
    write_pickle(net,'%d-%d.gz' % (net_count,lm_pct))
```

Ispis 7.6: Jednostavan automatizirani eksperiment. Za svaki od definiranih udjela sidara u mreži (5%, 10%, 20%, 33%) generira se 100 različitih mreža koje se sastoje od 100 do 300 čvorova s prosječno 9 susjeda. Svakoj mreži su pridodjeljeni opisani algoritmi za 1. i 2. fazu algoritma DV-hop koji se kasnije izvrše pokretanjem same simulacije. Nakon izvršenja svake simulacije podaci se spremaju u adekvatno imenovane datoteke. Definicija varijable ALGORITHMS je preskočena zbog sažetosti ispisa.



Slika 7.4: Prikaz topologije mreže unutar simulatora.



Slika 7.5: Grafički prikaz pogreške u estimaciji lokacije čvorova.

je prikazan u novom prozoru, kao na slici 7.4.

Interaktivna konzola može se iskoristiti za detaljnu analizu podataka dobivenih iz provedenih eksperimenata ili za povratak u pojedini trenutak izvršavanja eksperimenta.

```
# ispitaj regularne cvorove (nisu sidra) i ispisi njihove estimirane koordinate
In [19]: for node in net.nodes()[10:]:
   ....:   print node.memory.get('dvHop','Nije odredjena lokacija.')
[ 466.04  579.44]
[ 243.25   80.75]
[ 386.79   69.09]
[ 254.92  122.66]
[ 216.36   80.27]
Not localized.
[ 431.08  102.28]
[ 140.36  119.35]
etc...
In [20]: estimated = {}
In [21]: for node in net.nodes():
   ....:  if node.memory.has_key('dvHop'):
   ....:    estimated[node] = node.memory['dvHop']
In [22]: get_rms(net.pos,[estimated])
Out[22]: 32.19781563265385
In [23]: show_localized(net,[estimated],show_labels=False)
```

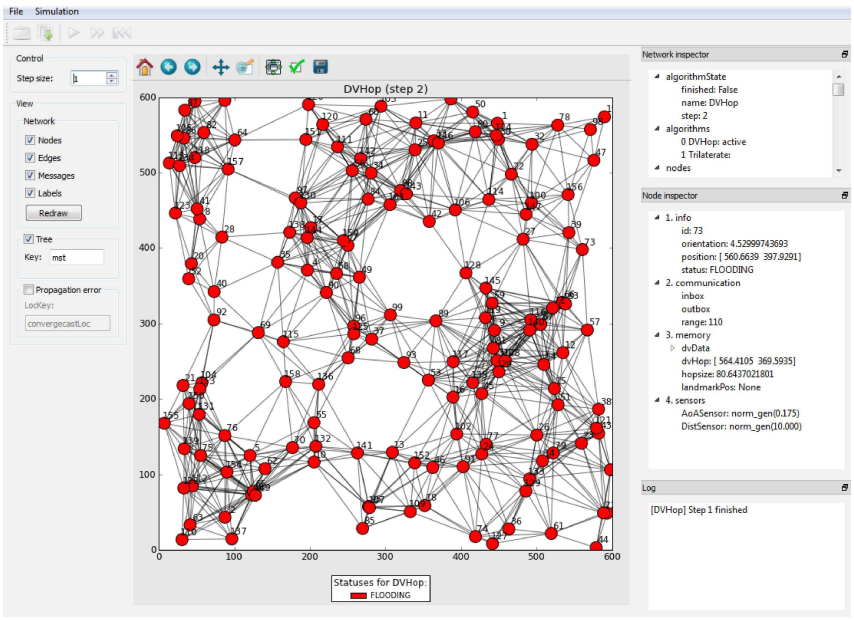
Ispis 7.7: Analiza memorije čvorova nakon provedene simulacije. Pod ključem `dvHop` u memoriji svih čvorova čija lokacija je uspješno estimirana nalaze se koordinate koje se mogu jednostavno očitati. Koristeći alate koje pruža Pymote knjižnica kao što je funkcija `get_rms` može se izračunati korijen srednje kvadratne pogreške estimiranih lokacija. Naposlijetku pogreške u estimaciji lokacije mogu biti i grafički vizualizirane koristeći funkciju `show_visualized` čiji rezultat je prikazan na slici 7.5. Pogreška je prikazana u formi crta koje povezuju estimirane lokacije čvorova, prikazane kružnicama, i točne lokacije čvorova.

Kao komplement načinima provođenja eksperimenata iz interaktivne konzole, Pymote uključuje i grafičko korisničko sučelje (engl. *graphical user interface – GUI*) za simulaciju i vizualizaciju izvođenja algoritama korak po korak. Prikaz sučelja dan je na slici 7.6. Koristeći ovo sučelje korisnik može:

- spremati i učitavati datoteke s podacima o pojedinoj mreži
- pokretati i nadzirati simulaciju korak po korak s izravnim vizualnim povratnim informacijama kao što su prosljeđivanje poruka, statusi čvorova i mreže i dr.
- provoditi inspekciju različitih objekata uključujući mrežu, čvorove i poruke jednostavnim klikom na njih
- prilagoditi prikaz mreže i njene topologije (bridovi, labele itd.)
- vizualizirati memorijske registre unutar čvorova koji se odnose na topologiju mreže kao što su stabla
- uvećati prikaz dijelova mreže
- spremati vizualizirane podatke u različitim slikovnim formatima.

U ovom poglavlju predstavljena je aktualna verzija Pymote simulatora za simulaciju i analizu raspodijeljenih algoritama u bežičnim mrežama. Planiran je daljnji razvoj simulatora u nekoliko različitih smjerova koji uključuju:

- dodavanje radnog okvira za još jednostavnije postavljanje složenih eksperimenata i automatiziranu analizu dobivenih podataka
- razvoj grafičkog korisničkog sučelja za kreiranje i postavljanje parametara mreža



Slika 7.6: Grafičko korisničko sučelje za simulaciju raspodijeljenih algoritama.

- implementaciju pojedinih raspodijeljenih algoritama za razne namjene koji predstavljaju najnovija postignuća u svojim područjima.

Programski kôd i poveznica na dokumentaciju projekta Pymote simulatora su dostupni na stranici https://github.com/darbula/pymote.

Poglavlje 8

Zaključak

Jedan od osnovnih preduvjeta u velikom dijelu primjena bežičnih mreža osjetila je poznavanje položaja čvorova. Čvorovi takvih mreža su vrlo ograničenih energetskih resursa, stoga je s aspekta potrošnje energije poželjno čim ravnomjernije raspodjeliti zadatke, te kooperativno i uz minimalnu komunikaciju doći do rješenja problema. Algoritmi koji implementiraju tu strategiju spadaju u razred raspodijeljenih algoritama, te u usporedbi s odgovarajućim centraliziranim algoritmima, osim u energetskom smislu, su u prednosti i s aspekta robusnosti jer njihovo uspješno funkcioniranje ne ovisi o raspoloživosti jednog centralnog čvora. Kako bi bilo moguće odrediti lokacije čvorova u globalnom odnosno apsolutnom koordinatnom sustavu, mreža mora sadržavati posebne, često strateški raspoređene, čvorove koji poznaju svoju lokaciju tzv. sidra, što predstavlja dodatno ograničenje primjene takvih algoritama. Stoga se u ovoj disertaciji razmatraju algoritmi koji ne ovise o prisutnosti sidara u mreži, a koji se ujedno izvršavaju raspodijeljeno, čim ravnomjernije koristeći resurse svih čvorova mreže.

Izvorni doprinos disertacije može se podijeliti u četiri glavna dijela. U prvom dijelu razvijena je metoda procjene kvalitete određivanja položaja čvorova korištenjem faktora geometrijskog slabljenja preciznosti dobivenog iz estimiranih lokacija. Osnovna hipoteza ovog dijela istraživanja je da se u algoritmima za određivanje lokacija može koristiti procjena točnosti estimacije lokacija koja se dobiva iz samih estimiranih koordinata. Iako se slična metoda s uspjehom koristi primjerice u sustavima globalnog pozicioniranja, u području bežičnih mreža osjetila do sada nije provedena analiza, niti je opisana metoda kojom bi se to učinilo. U radu se specifično opisuje metoda za mreže koje se koriste estimacijom azimuta. Rezultati simulacija pokazuju kako postoji korelacija između procijenjene i stvarne pogreške, posebice u mrežama s više čvorova i s manjom pogreškom estimacije azimuta.

Drugi dio doprinosa čini izvorni rapodijeljeni algoritam za određivanje položaja čvorova korištenjem estimiranih azimuta. Algoritam se zasniva na metodi spajanja lokalnih

koordinatnih sustava, a koristi se teorijom krutosti kako bi estimacija položaja bila dovoljno točna čak i u slučajevima malog prosječnog broja susjeda. Rezultati provedenih simulacija pokazuju kako je točnost ovog raspodijeljenog algoritma usporediva s centraliziranom verzijom, a u mrežama s nekonveksnom topologijom i nadmašuje centraliziranu verziju. Lokaciju je često potrebno estimirati korištenjem nedostatnih ili nedovoljno pouzdanih mjerenja što u slučajevima u kojima njihov broj i kvaliteta nije dovoljna može uzrokovati velike pogreške. Kako bi se iskoristila i dodatno potvrdila metoda iz prvog dijela doprinosa, ona je integrirana u izvorni algoritam u formi heuristike za detekciju i izbjegavanje korištenja podataka iz nedovoljno kvalitetno lokaliziranih dijelova mreže. Provedene simulacije pokazuju kako u mrežama s malom povezanošću, kod kojih može doći do pojave formacija koje su osjetljive na pogreške u estimaciji azimuta, metoda za procjenu kvalitete estimacije je sposobna detektirati takve slučajeve, što u konačnici rezultira kvalitetnijom estimacijom lokacija.

U trećem dijelu obavljena je eksperimentalna verifikacija izvornog raspodijeljenog algoritma za određivanje položaja u ispitnoj bežičnoj mreži osjetila. U sklopu verifikacije napravljena je analiza točnosti i komunikacijske složenosti. Prilikom implementacije algoritma bilo je potrebno učiniti određene modifikacije kako bi se omogućilo izvršavanje algoritma na memorijski i računalno vrlo ograničenoj platformi. Najveća modifikacija je implementacija mikroimunološkog algoritma za estimaciju lokacija u inicijalnim grozdovima, a koji se pokazao kao adekvatna zamjena za algoritam zasnovan na SVD-u. Dodatnu vrijednost eksperimentalne verifikacije čini razvoj jedinstvenog osjetila za određivanje smjera dolaska infracrvenog signala. U sklopu razvoja osjetila razvijena je i metoda za automatizirano ispitivanje i umjeravanje, te algoritam za estimaciju azimuta dolaznog signala. Rezultati ispitivanja točnosti estimacije azimuta za 16 izrađenih i umjerenih osjetila pokazuju kako je standardna devijacija pogreške estimacije azimuta između 0,6° i 1,4°, dok je u realnim primjenama nešto veća zbog pojave refleksija.

Konačno, četvrti dio doprinosa čini programski simulator raspodijeljenih algoritama u bežičnim mrežama osjetila. Simulator je bio preduvjet provedenog istraživanja te je omogućio implementaciju i ispitivanje složenih strategija i metoda. Kako simulator funkcionira na aplikacijskoj razini, njegovim korištenjem moguće je vrlo brzo analizirati ponašanje algoritma u raspodijeljenom računalnom okruženju, otkriti i ispraviti moguće uzroke zastoja, te dobiti preliminarne rezultate za ispitivani algoritam. Dodatna mogućnost koju pruža Python kao odabrani programski jezik, je jednostavna integracija knjižnica napisanih u programskim jezicima C i C++ (koji se koriste u mnogim platformama) unutar samog simulatora. Tako je, primjerice, u sklopu istraživanja unutar simulatora i u konačnoj implementaciji na ispitnoj mreži korištena identična knjižnica za mikroimunološki algoritam što je omogućilo kvalitetnije ispitivanje algoritma i brži ra-

zvoj. Konačna verifikacija simulatora i odabrane metode razvoja dobivena je uspješnom usporedbom rezultata algoritma u simulatoru i u ispitnoj mreži.

Potencijalne smjernice budućeg rada uključuju dodatno istraživanje metoda procjene pogreške, s naglaskom na analitički opis veze između broja čvorova, pogreške mjerenja azimuta, te korelacije između procjenjene i prave pogreške estimacije položaja. Također se planira ispitati ista metoda u algoritmima koji se služe mjerenjima udaljenosti među čvorovima. Tijekom implementacije algoritma u ispitnoj mreži pojavila se potreba za rješavanjem problema raspodijeljene selekcije susjeda s očuvanjem krutosti cjelokupne mreže, a koji također zavrijeđuje dodatno istraživanje. Jedan od planiranih zadataka je i ispitivanje algoritma na platformi koja bi omogućila implementaciju heuristike i analitičkog rješenja zasnovanog na SVD-u, dok se na trenutno odabranoj platformi može učiniti i dodatni napredak optimizacijom mikroimunološkog algoritma, posebice s aspekta ograničenja prostora pretraživanja, te implementacije aritmetike s nepomičnim zarezom. U planu je i nastavak rada na osjetilu za mjerenje azimuta, te primjena na estimaciju lokacija mobilnih čvorova.

Literatura

[1] Moore, D., Leonard, J., Rus, D., Teller, S., "Robust distributed network localization with noisy range measurements", in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, ser. SenSys '04. New York, NY, USA: ACM, 2004, str. 50–61, dostupno na: http://doi.acm.org/10.1145/1031495.1031502

[2] Shang, Y., Ruml, W., Fromherz, M. P. J., "Positioning using local maps", Ad Hoc Netw., Vol. 4, No. 2, Mar. 2006, str. 240–253, dostupno na: http://dx.doi.org/10.1016/j.adhoc.2004.06.001

[3] Cucuringu, M., Lipman, Y., Singer, A., "Sensor network localization by eigenvector synchronization over the euclidean group", ACM Trans. Sen. Netw., Vol. 8, No. 3, Aug. 2012, str. 19:1–19:42, dostupno na: http://doi.acm.org/10.1145/2240092.2240093

[4] Kwon, O.-H., Song, H.-J., "Localization through map stitching in wireless sensor networks", IEEE Transactions on Parallel and Distributed Systems, Vol. 19, No. 1, Jan. 2008, str. 93–105.

[5] Wang, X., Liu, Y., Yang, Z., Lu, K., Luo, J., "Robust component-based localization in sparse networks", IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 5, May 2014, str. 1317–1327.

[6] Arbula, D., Raspodijeljeni algoritam za lokalizaciju u neusidrenoj mreži određivanjem smjera dolaska signala, Magistarski rad. Fakultet elektrotehnike i računarstva, Zagreb, 2008.

[7] Kwon, O.-H., Song, H.-J., Park, S., "Anchor-free localization through flip-error-resistant map stitching in wireless sensor network", Parallel and Distributed Systems, IEEE Transactions on, Vol. 21, No. 11, Nov. 2010, str. 1644 –1657.

[8] Aspnes, J., Eren, T., Goldenberg, D., Morse, A., Whiteley, W., Yang, Y., Anderson, B., Belhumeur, P., "A theory of network localization", IEEE Transactions on Mobile Computing, Vol. 5, No. 12, Dec. 2006, str. 1663–1678.

[9] Torrieri, D., "Statistical theory of passive location systems", Aerospace and Electronic Systems, IEEE Tansactions on, Vol. 20, No. 2, Mar. 1984, str. 183–198.

[10] Hsu, J., "Insect-eye camera offers wide-angle vision for tiny drones", IEEE Spectrum, Jan. 2013, dostupno na: http://spectrum.ieee.org/robotics/robotics-hardware/insecteye-camera-offers-wideangle-vision-for-tiny-drones/

[11] Reichenbach, F., Born, A., Timmermann, D., Bill, R., "A distributed linear least squares method for precise localization with low complexity in wireless sensor networks", in Proceedings of the Second IEEE International Conference on Distributed Computing in Sensor Systems, ser. DCOSS'06. Berlin, Heidelberg: Springer-Verlag, 2006, str. 514–528, dostupno na: http://dx.doi.org/10.1007/11776178_31

[12] Glumac, S., Kovačić, Z., "Microimmune algorithm for solving inverse kinematics of redundant robots", in 2013 IEEE International Conference on Systems, Man, and Cybernetics, 2013.

[13] Priyantha, N. B., Chakraborty, A., Balakrishnan, H., "The cricket location-support system", in 6th ACM International Conference on Mobile Computing and Networking (ACM MOBICOM), Boston, MA, August 2000, Aug. 2000.

[14] Mao, G., Fidan, B., Anderson, B. D. O., "Wireless sensor network localization techniques", Computer Networks, Vol. 51, No. 10, Jul. 2007, str. 2529–2553, dostupno na: http://www.sciencedirect.com/science/article/pii/S1389128606003227

[15] Savvides, A., Han, C., Strivastava, M. B., "Dynamic fine-grained localization in ad-hoc networks of sensors", in Proceedings of MOBICOM'01, Rome, Italy, Jul. 2001.

[16] Stojmenovic, I., (ur.), Handbook of Sensor Networks: Algorithms and Architectures, ser. Series on Parallel and Distributed Computing. John Wiley & Sons, Inc., 2005.

[17] Romer, K., "The lighthouse location system for smart dust", in Proceedings of MobiSys 2003 (ACM/USENIX Conference on Mobile Systems, Applications and Services), 2003.

[18] Priyantha, N. B., Miu, A. K. L., Balakrishnan, H., Teller, S., "The cricket compass for context aware mobile applications", in 7th ACMConf. Mobile Computing and Networking (MOBICOM), Jul. 2001.

[19] Nasipuri, A., Li, K., "A directionality based location discovery scheme for wireless sensor networks", in WSNA'02 September 28, 2002, Atlanta, Georgia, USA. ACM, Sep. 2002.

[20] Yang, Z., Wu, C., Liu, Y., "Locating in fingerprint space: wireless indoor localization with little human intervention", in Proceedings of the 18th annual international conference on Mobile computing and networking, ser. Mobicom '12. Istanbul, Turkey: ACM, 2012, str. 269–280.

[21] Eren, T., Whiteley, W., Belhumeur, P. N., "A theoretical analysis of the conditions for unambiguous node localization in sensor networks", Department of Computer Science, Columbia University, Tech. Rep., 2004.

[22] Jackson, B., Jordan, T., "Connected rigidity matroids and unique realizations of graphs", Combinatorial Theory, Vol. 94, 2005, str. 1–29.

[23] Eren, T., "Cooperative localization in wireless ad hoc and sensor networks using hybrid distance and bearing (angle of arrival) measurements", EURASIP Journal on Wireless Communications and Networking, Vol. 2011, No. 1, Aug. 2011, str. 72, dostupno na: http://jwcn.eurasipjournals.com/content/2011/1/72/abstract

[24] Mao, G., Fidan, B., (ur.), Localization Algorithms and Strategies for Wireless Sensor Networks: Monitoring and Surveillance Techniques for Target Tracking. IGI Global, May 2009, dostupno na: http://www.igi-global.com/book/localization-algorithms-strategies-wireless-sensor/700

[25] Aksu, H., Aksoy, D., Korpeoglu, I., "A study of localization metrics: Evaluation of position errors in wireless sensor networks", Computer Networks, Vol. 55, No. 15, 2011, str. 3562 – 3577.

[26] Chen, J., Yao, K., Hudson, R., "Source localization and beamforming", Signal Processing Magazine, IEEE, Vol. 19, No. 2, Mar. 2002, str. 30 –39.

[27] Savvides, A., Park, H., Srivastava, M. B., "The bits and flops of the n-hop multilateration primitive for node localization problems", in WSNA'02 September 28, 2002, Atlanta, Georgia, USA, Sep. 2002.

[28] Nasipuri, A., el Najjar, R., "Experimental evaluation of an angle based indoor localization system", in 2006 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, Apr. 2006, str. 1–9.

[29] Niculescu, D., Nath, B., "Ad hoc positioning system (APS) using AOA", in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, Vol. 3, Apr. 2003, str. 1734–1743.

[30] Yang, Z., Liu, Y., "Quality of trilateration: Confidence-based iterative localization", IEEE Transactions on Parallel and Distributed Systems, Vol. 21, No. 5, 2010, str. 631–640.

[31] Shao, H.-J., Zhang, X.-P., Wang, Z., "Efficient closed-form algorithms for AOA based self-localization of sensor nodes using auxiliary variables", IEEE Transactions on Signal Processing, Vol. 62, No. 10, May 2014, str. 2580–2594.

[32] Borg, I., Groenen, P. J. F., Modern Multidimensional Scaling: Theory and Applications. Springer, 2005.

[33] Biswas, P., Ye, Y., "Semidefinite programming for ad hoc wireless sensor network localization", in Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on, Apr. 2004, str. 46 – 54.

[34] Niculescu, D., Nath, B., "Ad hoc positioning system (APS)", in Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE, Vol. 5, Nov. 2001, str. 2926–2931.

[35] Kannan, A., Mao, G., Vucetic, B., "Simulated annealing based wireless sensor network localization with flip ambiguity mitigation", in Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd, Vol. 2, May 2006, str. 1022 –1026.

[36] Vanheel, F., Verhaevert, J., Laermans, E., Moerman, I., Demeester, P., "Automated linear regression tools improve RSSI WSN localization in multipath indoor environment", EURASIP Journal on Wireless Communications and Networking, Vol. 38, 2011.

[37] Amundson, I., Sallai, J., Koutsoukos, X., Ledeczi, A., Maroti, M., "RF angle of arrival-based node localisation", Int. J. Sen. Netw., Vol. 9, No. 3/4, May 2011, str. 209–224, dostupno na: http://dx.doi.org/10.1504/IJSNET.2011.040241

[38] Jiang, J.-R., Lin, C.-M., Lin, F.-Y., Huang, S.-T., "ALRD: AoA localization with RSSI differences of directional antennas for wireless sensor networks", International Journal of Distributed Sensor Networks, Vol. 2013, Mar. 2013, dostupno na: http://www.hindawi.com/journals/ijdsn/2013/529489/abs/

[39] Katz, B., Gaertler, M., Wagner, D., "Maximum rigid components as means for direction-based localization in sensor networks", in SOFSEM 2007: Theory and Practice of Computer Science, ser. Lecture Notes in Computer Science, Leeuwen, J. v., Italiano, G. F., Hoek, W. v. d., Meinel, C., Sack, H., Plášil, F., (ur.). Springer Berlin Heidelberg, Jan. 2007, No. 4362, str. 330–341, dostupno na: http://link.springer.com/chapter/10.1007/978-3-540-69507-3_27

[40] Tang, B., Zhu, X., Subramanian, A., Gao, J., "DAL: A distributed localization in sensor networks using local angle measurement", in Proceedings of 18th Internatonal Conference on Computer Communications and Networks, 2009. ICCCN 2009, Aug. 2009, str. 1–6.

[41] Rong, P., Sichitiu, M., "Angle of arrival localization for wireless sensor networks", in Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on, Vol. 1, Sep. 2006, str. 374–382.

[42] Di Stefano, G., Petricola, A., "A distributed AOA based localization algorithm for wireless sensor networks", Journal of Computers, 2008.

[43] Zhu, G., Hu, J., "Distributed network localization using angle-of-arrival information part II: Discrete-time algorithm and error analysis", in American Control Conference (ACC), 2013, Jun. 2013, str. 1000–1005.

[44] Zhu, G., Hu, J., "A distributed continuous-time algorithm for network localization using angle-of-arrival information", Automatica, Vol. 50, No. 1, Jan. 2014, str. 53–63, dostupno na: http://dx.doi.org/10.1016/j.automatica.2013.09.033

[45] Saad, C., Benslimane, A., Konig, J.-C., "AT-angle: A distributed method for localization using angles in sensor networks", in IEEE Symposium on Computers and Communications, 2008. ISCC 2008, Jul. 2008, str. 1190–1195.

[46] Champ, J., Boudet, V., "ADNL-angle: Accurate distributed node localization for wireless sensor networks with angle of arrival information", in Ad-Hoc, Mobile and Wireless Networks, ser. Lecture Notes in Computer Science, Nikolaidis, I., Wu, K., (ur.). Springer Berlin Heidelberg, Jan. 2010, No. 6288, str. 177–190, dostupno na: http://link.springer.com/chapter/10.1007/978-3-642-14785-2_14

[47] Ash, J. N., Potter, L. C., "Robust system multiangulation using subspace methods", in IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks. Cambridge, Massachusetts, USA: ACM Press, 2007, str. 61–68.

[48] Shang, Y., Ruml, W., "Improved MDS-based localization", in INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, Vol. 4, Mar. 2004, str. 2640–2651 vol.4.

[49] Kwon, O.-H., Song, H.-J., Park, S., "The effects of stitching orders in patch-and-stitch WSN localization algorithms", IEEE Trans. Parallel Distrib. Syst., Vol. 20, No. 9, Sep. 2009, str. 1380–1391, dostupno na: http://dx.doi.org/10.1109/TPDS.2008.226

[50] Bancroft, S., "An algebraic solution of the GPS equations", IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-21, No. 1, Jan. 1985, str. 56–59.

[51] Chen, C.-S., Chiu, Y.-J., Lee, C.-T., Lin, J.-M., "Calculation of weighted geometric dilution of precision", Journal of Applied Mathematics, Vol. 2013, Oct. 2013, dostupno na: http://www.hindawi.com/journals/jam/2013/953048/abs/

[52] "NAVSTAR GPS user equipment introduction", dostupno na: http://www.navcen.uscg.gov/pubs/gps/gpsuser/gpsuser.pdf 1996.

[53] Dempster, A., "Dilution of precision in angle-of-arrival positioning systems", Electronics Letters, Vol. 42, No. 5, Mar. 2006, str. 291–292.

[54] Blewitt, G., "Basics of the GPS technique: Observation equations", in Geodetic Applications for GPS. Nordic Geodetic Commission, Sweden, 1997, str. 10–54.

[55] Chang, C., Sahai, A., "Cramer-rao-type bounds for localization", EURASIP Journal on Applied Signal Processing, Vol. 2006, 2006, str. 1 – 13.

[56] Ash, J., Moses, R., "On the relative and absolute positioning errors in self-localization systems", IEEE Transactions on Signal Processing, Vol. 56, No. 11, Nov. 2008, str. 5668–5679.

[57] Savvides, A., Garber, W., Adlakha, S., Moses, R., Srivastava, M., "On the error characteristics of multihop node localization in ad-hoc sensor networks", in Proceedings of the Second International Workshop on Information Processing in Sensor Networks (IPSN'03), Paolo Alto, California, Apr. 2003, str. 317 – 332.

[58] Patwari, N., Ash, J., Kyperountas, S., Hero, I., A.O., Moses, R., Correal, N., "Locating the nodes: cooperative localization in wireless sensor networks", Signal Processing Magazine, IEEE, Vol. 22, No. 4, Jul. 2005, str. 54–69.

[59] Moses, R., Patterson, R., "Self calibration of sensor networks", in Proceedings of SPIE, ser. Unattended Ground Sensor Technologies and Applications IV, Vol. 474, Apr. 2002.

[60] Chang, C., Sahai, A., "Estimation bounds for localization", in Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on, Oct. 2004, str. 415–424.

[61] Shames, I., Bishop, A., Anderson, B., "Analysis of noisy bearing-only network localization", IEEE Transactions on Automatic Control, Vol. 58, No. 1, Jan. 2013, str. 247–252.

[62] Santoro, N., Design and analysis of distributed algorithms, ser. Series on Parallel and Distributed Computing. John Wiley & Sons, Inc., 2007.

[63] Fang, J., Duncan, D., Morse, A., "Sequential localization with inaccurate measurements", in American Control Conference, 2009. ACC '09., Jun. 2009, str. 1970–1975.

[64] Bron, C., Kerbosch, J., "Algorithm 457: Finding all cliques of an undirected graph", Commun. ACM, Vol. 16, No. 9, Sep. 1973, str. 575–577, dostupno na: http://doi.acm.org/10.1145/362342.362367

[65] Anderson, B., Eren, T., Whiteley, W., Morse, A., Belhumeur, P., "Merging globally rigid formations of mobile autonomous agents", in Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on, Jul. 2004, str. 1258–1259.

[66] Arbula, D., "Distributed algorithm for anchor-free network localization using angle of arrival", in IEEE International Symposium on Industrial Electronics, 2008. ISIE 2008, Jun. 2008, str. 792–797.

[67] Eren, T., Whiteley, W., Morse, A. S., Belhumeur, P. N., Anderson, B., "Sensor and network topologies of formations with direction, bearing and angle information between agents", in Proceedings of the 42nd IEEE Conference on Decision and Control, Dec. 2003, str. 3064–3069.

[68] Horn, B., Hilden, H., Negahdaripour, S., "Closed-form solution of absolute orientation using orthonormal matrices", Journal of the Optical Society of America A, Vol. 5, 1988, str. 1127.

[69] Gudger, K., "AVR libc documentation", dostupno na: http://www.nongnu.org/avr-libc/ 2014.

[70] Song, Y. M., Xie, Y., Malyarchuk, V., Xiao, J., Jung, I., Choi, K.-J., Liu, Z., Park, H., Lu, C., Kim, R.-H., Li, R., Crozier, K. B., Huang, Y., Rogers, J. A., "Digital cameras with designs inspired by the arthropod eye", Nature, Vol. 497, No. 7447, May 2013, str. 95–99, dostupno na: http://www.nature.com/nature/journal/v497/n7447/full/nature12083.html

[71] Vishay, "VEMT2023slx01 IR phototransistor, NPN, SMD, technical data sheet", dostupno na: http://www.vishay.com/docs/84166/vemt2023slx01.pdf 2012.

[72] Vishay, "VSMB2943slx01 IR emitter, SMD, technical data sheet", dostupno na: http://www.vishay.com/docs/83479/vsmb2943slx01.pdf 2012.

[73] Dužanec, D., Kovačić, Z., "Performance analysis-based GA parameter selection and increase of uGA accuracy by gradual contraction of solution space", in IEEE International Conference on Industrial Technology, 2009. ICIT 2009, Feb. 2009, str. 1–7.

[74] Arbula, D., Lenac, K., "Pymote: High level python library for event-based simulation and evaluation of distributed algorithms", International Journal of Distributed Sensor Networks, Vol. 2013, Mar. 2013, dostupno na: http://www.hindawi.com/journals/ijdsn/2013/797354/abs/

[75] Downard, I. T., "Simulating sensor networks in NS-2", Naval Research Laboratory, Tech. Rep., 2004.

[76] Varga, A., "The OMNeT++ discrete event simulation system", in Proceedings of the European Simulation Multiconference (ESM'2001), Prague, Jun. 2001.

[77] Boulis, A., "Castalia: Revealing pitfalls in designing distributed algorithms in WSN", in Proceedings of the 5th International Conference on Embedded Networked Sensor Systems, ser. SenSys '07. New York, NY, USA: ACM, 2007, str. 407–408, dostupno na: http://doi.acm.org/10.1145/1322263.1322318

[78] Levis, P., Lee, N., Welsh, M., Culler, D., "TOSSIM: Accurate and scalable simulation of entire TinyOS applications", in Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, ser. SenSys '03. New York, NY, USA: ACM, 2003, str. 126–137, dostupno na: http://doi.acm.org/10.1145/958491.958506

[79] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., Culler, D., "TinyOS: An operating system for sensor networks", in Ambient Intelligence, Weber, W., Rabaey, J.,

Aarts, E., (ur.). Springer Berlin Heidelberg, 2005, str. 115–148, dostupno na: http://dx.doi.org/10.1007/3-540-27139-2_7

[80] Egea-Lopez, E., Vales-Alonso, J., Martinez-Sala, A. S., Pavon-Marino, P., Garcia-Haro, J., "Simulation tools for wireless sensor networks", in Summer Simulation Multiconference-SPECTS, Vol. 2005, 2005.

[81] Mehta, S., Sulatan, N., Kwak, K. S., "Network and system simulation tools for next generation networks: a case study", Modelling, Simulation and Identification, 2010.

[82] Sundani, H., Li, H., Devabhaktuni, V. K., Alam, M., Bhattacharya, P., "Wireless sensor network simulators: A survey and comparisons", International Journal Of Computer Networks (IJCN), Vol. 2, No. 5, 2011, str. 249–265.

[83] Shu, L., Hauswirth, M., Chao, H.-C., Chen, M., Zhang, Y., "NetTopo: A framework of simulation and visualization for wireless sensor networks", Ad Hoc Networks, 2011, str. 799–820.

[84] Kroeller, A., Pfisterer, D., Buschmann, C., Fekete, S., Fischer, S., "Shawn: A new approach to simulating wireless sensor networks", in Proceedings of the Design, Analysis, and Simulation of Distributed Systems (DASD 05), San Diego, 2005, dostupno na: https://github.com/itm/shawn

[85] Pérez, F., Granger, B. E., "IPython: A system for interactive scientific computing, computing", Science and Engineering, Vol. 9, No. 3, Jun. 2007, str. 21–29, dostupno na: http://ipython.org

[86] Jones, E., Oliphant, T., Peterson, P. *et al.*, "SciPy: Open source scientific tools for python", 2001, dostupno na: http://www.scipy.org/

[87] Hagberg, A. A., Schult, D. A., Swart, P. J., "Exploring network structure, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science Conference (SciPy2008), Pasadena, CA USA, Aug. 2008, str. 11–15.

[88] Niculescu, D., Nath, B., "DV based positioning in ad hoc networks", Telecommunication Systems, Vol. 22, No. 1-4, Jan. 2003, str. 267–280, dostupno na: http://link.springer.com/article/10.1023/A%3A1023403323460

Prilozi

P1 Robusna angulacija

U ovom prilogu opisana je metoda estimacije lokacija koja je korištena u simulacijama, a zasniva se na algoritmu RAST opisanom u [47].

Izmjerene vrijednosti azimuta mogu se napisati kao jedinični vektori rotirani za 90° od pravog azimuta.

$$\boldsymbol{u}_{ij} = \begin{bmatrix} \sin\theta_{ij} \\ \\ -\cos\theta_{ij} \end{bmatrix} \tag{P.1}$$

gdje je θ_{ij} azimut čvora j izmjerenog osjetilom na čvoru i u globalnom koordinatnom sustavu. S obzirom da su orijentacije čvorova u globalnom koordinatnom sustavu nepoznate, čvorovi mogu mjeriti samo azimut u relativnom koordinatnom sustavu prijamnog čvora ϕ_{ij}. Prema tome, potrebno je estimirati i globalnu orijentaciju prijamnog čvora α_i kako bi se izračunao globalni azimut θ_{ij} korištenjem sljedeće jednadžbe:

$$\theta_{ij} = \phi_{ij} + \alpha_i. \tag{P.2}$$

P1.1 Estimacije orijentacija

U cilju pojednostavljenja modulo 2π aritmetike, elementi matrice $\boldsymbol{\Psi}$, odnosno $\alpha_i - \alpha_j$ mogu se predstaviti u eksponencijalnom obliku kao faze točaka u kompleksnoj ravnini $e^{(\alpha_i - \alpha_j)i}$ pa se matrica Ψ transformira u matricu B:

$$\boldsymbol{B} = e^{i\boldsymbol{\Psi}} + 2I. \tag{P.3}$$

Kako osjetilo za mjerenje azimuta ima ograničen domet, pojedina mjerenja u matrici $\boldsymbol{\Phi}$ mogu izostati. U tom slučaju potrebno je estimirati elemente matrice $\boldsymbol{B}$ tako što će se pomnožiti odgovarajuće vrijednosti iste matrice koje odgovaraju bridovima koji se

nalaze na najkraćem putu $\mathcal{P}$ između čvorova čija mjerenja nedostaju:

$$\boldsymbol{B}_{ij} = e^{i(\alpha_i - \alpha_j)} = e^{i(\alpha_i - \alpha_{a_1})} \cdots e^{i(\alpha_{a_p} - \alpha_j)} = \prod_{(k,l)\in\mathcal{P}} B_{mn}. \tag{P.4}$$

U slučaju kada se estimiraju lokacije u mreži koja se sastoji od čvorova koji su svi susjedi jednog čvora, najkraći put je ili izravan ili prolazi upravo kroz taj čvor.

Elementi matrice $\boldsymbol{B}$ su $\boldsymbol{B}_{ij} = e^{i(\alpha_i - \alpha_j)}$, pa ako se definira da je $a(\alpha) = [e^{i\alpha_1}, \ldots, e^{\alpha_n}]^T$ može se napisati:

$$\boldsymbol{B} = a(\alpha)a(\alpha)^* \tag{P.5}$$

Cilj je pronaći takav skup orijentacija $a(\alpha)$ da $a(\alpha)a(\alpha)^*$ bude najsličniji $\boldsymbol{B}$ odnosno razlikama orijentacija dobivenih iz mjerenja. Ova metoda se naziva aproksimacija nižeg ranga (engl. *low-rank approximation*), pri čemu je $a(\alpha)a(\alpha)^*$ aproksimirajuća matrica, a $\boldsymbol{B}$ je podatkovna matrica. Kako rang matrice $a(\alpha)a(\alpha)^*$ iznosi 1, ovaj slučaj spada pod aproksimaciju ranga 1. Aproksimacija se obavlja korištenjem rastava po singularnim vrijednostima matrice $\boldsymbol{B}$, pri čemu je rješenje desni singularni vektor koji odgovara najvećoj singularnoj vrijednosti. Nakon što su estimirane vrijednosti orijentacija, mjerenja se mogu izraziti u globalno orijentiranom koordinatnom sustavu korištenjem jednadžbe P.2.

Korištenjem činjenice da bi se mjerenja azimuta, pod pretpostavkom izostanka mjernih pogrešaka i pod pretpostavkom identičnih orijentacija čvorova, trebale razlikovati točno za π, razlika između orijentacija može se predstaviti sljedećom jednadžbom:

$$\alpha_i - \alpha_j = \phi_{ji} - \phi_{ij} + \pi. \tag{P.6}$$

Za sve parove mjerenja organizirane u matricu $\Phi = \{\phi_{ij}\}$ sustav jednadžbi u matričnom obliku glasi:

$$\Psi = \Phi^T - \Phi + \pi 1_n 1_n^T \tag{P.7}$$

P1.2 Estimacije lokacija

S obzirom da je vektor $\boldsymbol{u}_{ij}$ orijentiran pod kutom od 90° prema pravom azimutu, odnosno razlici lokacija čvorova j i i, vrijedi sljedeća jednadžba:

$$\boldsymbol{u}_{ij}^T(\boldsymbol{p}_j - \boldsymbol{p}_i) = 0 \tag{P.8}$$

gdje su $\boldsymbol{p}_j$ i $\boldsymbol{p}_i$ lokacije čvorova.

Sustav sastavljen od jednadžbi (P.8), koje reprezentiraju mjerenja azimuta između

svih parova susjednih čvorova, može se napisati u matričnom obliku:

$$\boldsymbol{U}^T\boldsymbol{K}\boldsymbol{p} = 0 \tag{P.9}$$

gdje je $\boldsymbol{U}$ blok dijagonalna matrica, sastavljena od vektora $\boldsymbol{u}_{ij}$ svih parova susjeda, a $\boldsymbol{p}$ vektor stupac koordinata svih n čvorova. Matrica $\boldsymbol{K}$ je posebno konstruirana matrica sastavljena od elemenata 0, -1 i 1, na način da se umnoškom s vektorom lokacija kreiraju razlike lokacija na odgovarajućim pozicijama rezultantne matrice.

Rješenje problema relativne lokalizacije je partikularno rješenje linearnog homogenog sustava (P.9), pri čemu je, uz dovoljan broj mjerenja, dimenzija nul-prostora matrice $\boldsymbol{U}^T\boldsymbol{K}$ jednaka 3. Dimenzije odgovaraju kongruentnim transformacijama paralelno krutih formacija. Jedna dimenzija se može predstaviti vektorom translacije po x-osi $\boldsymbol{v}_x = [1\ 0\ 1\ 0\ \ldots]^T$, druga vektorom translacije po y-osi $\boldsymbol{v}_y = [0\ 1\ 0\ 1\ \ldots]^T$, a treća dimenzija predstavlja vektor skaliranja, koji je jednak upravo vektoru relativnih lokacija $\boldsymbol{p}$.

Dodavanjem vektora $\boldsymbol{v}_x$ i $\boldsymbol{v}_y$ matrici $\boldsymbol{U}^T\boldsymbol{K}$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{U}^T\boldsymbol{K} \\ \boldsymbol{v}_x^T \\ \boldsymbol{v}_y^T \end{bmatrix} \tag{P.10}$$

reducira se nulprostor te vrijedi $\mathcal{N}(\boldsymbol{A}) = \text{span}(\boldsymbol{p})$. Prema tome sustav:

$$\boldsymbol{A}\boldsymbol{p} = 0 \tag{P.11}$$

je rješiv koristeći se rastavom singularnih vrijednosti $\boldsymbol{A} = \boldsymbol{U}_A\boldsymbol{\Sigma}_A\boldsymbol{V}_A^T$, pri čemu je najbolja estimacija vektora relativnih lokacija $\boldsymbol{p}$ singularni vektor, koji odgovara najmanjoj singularnoj vrijednosti matrice $\boldsymbol{A}$

$$\hat{\boldsymbol{p}} = \boldsymbol{V}_A^{(2n)}. \tag{P.12}$$

P2 Spajanje podgrozdova

Zadatak spajanja podgrozdova dijeli se na tri slučaja: kada između podgrozdova koji se spajaju postoji jedan, dva odnosno više od dva zajednička čvora.

U slučaju kada postoji samo jedan zajednički čvor, matrica rotacije određuje se estimacijom kuta φ na način da se usklade azimuti čvorova koji su incidentni sa zajedničkim čvorom u njegovom lokalnom koordinatnom sustavu. Kako bi se podgrozdovi uskladili rotacijom, vektor lokacija čvorova iz oba podgrozda množi se odgovarajućom matricom rotacije $\mathbf{R}$:

$$\boldsymbol{R} = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{bmatrix}. \tag{P.13}$$

pri čemu je s φ označen kut između izmjerenog i estimiranog azimuta za čvorove koji su incidentni sa zajedničkim čvorom u jednom i u drugom sustavu. Skaliranje je u ovom slučaju suvišno, a vektor translacije računa se kao i za ostale slučajeve. Ukoliko postoje dva zajednička čvora, matrica rotacije računa se na sličan način, s razlikom da je φ kut između vektora koji povezuju dva zajednička čvora u jednom i u drugom sustavu.

Za slučaj s više od dva zajednička čvora koristi se metoda opisana u [68]:

- Pronaći centroide n_z zajedničkih čvorova za oba podgrozda

$$\bar{p}_s = \frac{1}{n_z}\sum_{i=1}^{n_z} p_{s,i} \quad \bar{p}_d = \frac{1}{n_z}\sum_{i=1}^{n_z} p_{d,i} \tag{P.14}$$

  gdje su p_s i p_d lokacije zajedničkih čvorova u izvorišnom odnosno odredišnom koordinatnom sustavu.
- Translatirati sve lokacije tako da su definirane u odnosu na centroide

$$\bar{p}'_{s,i} = p_{s,i} - \bar{p}_s \quad \bar{p}'_{d,i} = p_{d,i} - \bar{p}_d \tag{P.15}$$

- Izračunati matricu $\boldsymbol{M}$ veličine 2×2

$$\boldsymbol{M} = \sum_{i=1}^{n_z} p'_{d,i}(p'_{s,i})^T. \tag{P.16}$$

- Za matricu $\boldsymbol{M}^T\boldsymbol{M}$ izračunati svojstvene vrijednosti λ_1,λ_2 i svojstvene vektore $\hat{u}_1$,$\hat{u}_2$. Matrica rotacije se tada može izračunati kao

$$\boldsymbol{R} = \boldsymbol{M}\left(\frac{1}{\sqrt{\lambda_1}}u_1\hat{u}_1 + \frac{1}{\sqrt{\lambda_2}}u_2\hat{u}_2\right). \tag{P.17}$$

Estimacija **faktora skaliranja** ξ za slučajeve kada postoji više od jednog zajedničkog čvora obavlja se koristeći sljedeću jednadžbu

$$\xi = \sqrt{\frac{\sum_{i=1}^{n_z} \|p'_{d,i}\|^2}{\sum_{i=1}^{n_z} \|p'_{s,i}\|^2}}. \tag{P.18}$$

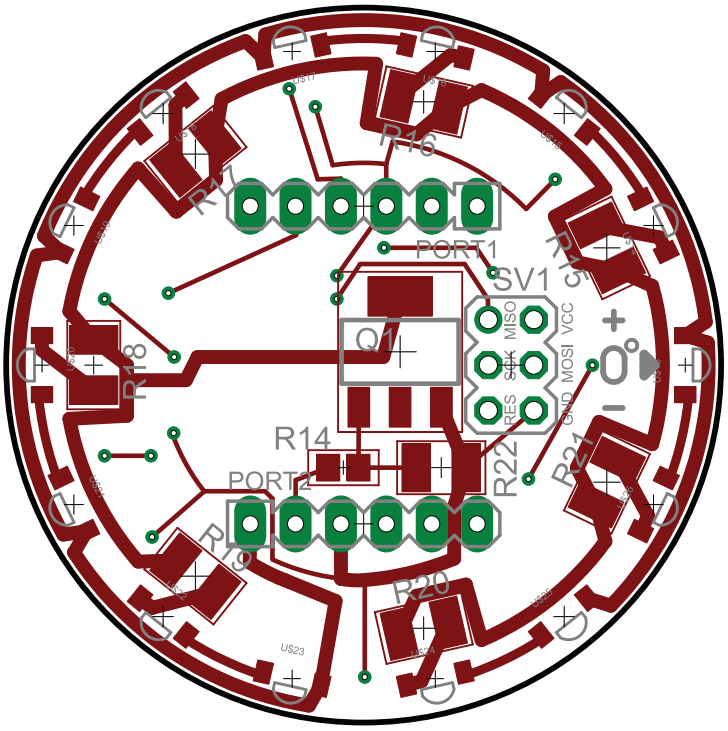
Procedure estimacije **vektora translacije** $\vec{t}$ za sva tri slučaja su iste. Potrebno je translatirati prethodno rotiranu i skaliranu centroidu zajedničkih čvorova izvorišnog podgrozda tako da se uskladi s centroidom zajedničkih čvorova odredišnog podgrozda

$$\vec{t} = \bar{p}_d - \xi \boldsymbol{R} \bar{p}_s. \tag{P.19}$$

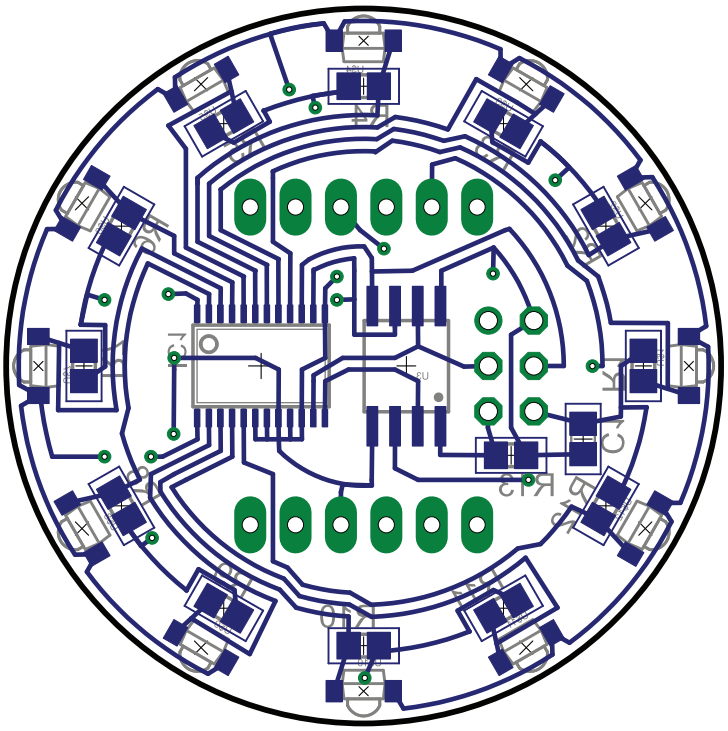
Naposljetku, lokacije čvorova u odredišnom podgrozdu računaju se koristeći sljedeći izraz:

$$p_d = \vec{t} + \xi \boldsymbol{R} p_s. \tag{P.20}$$

P3 Osjetilo za mjerenje azimuta

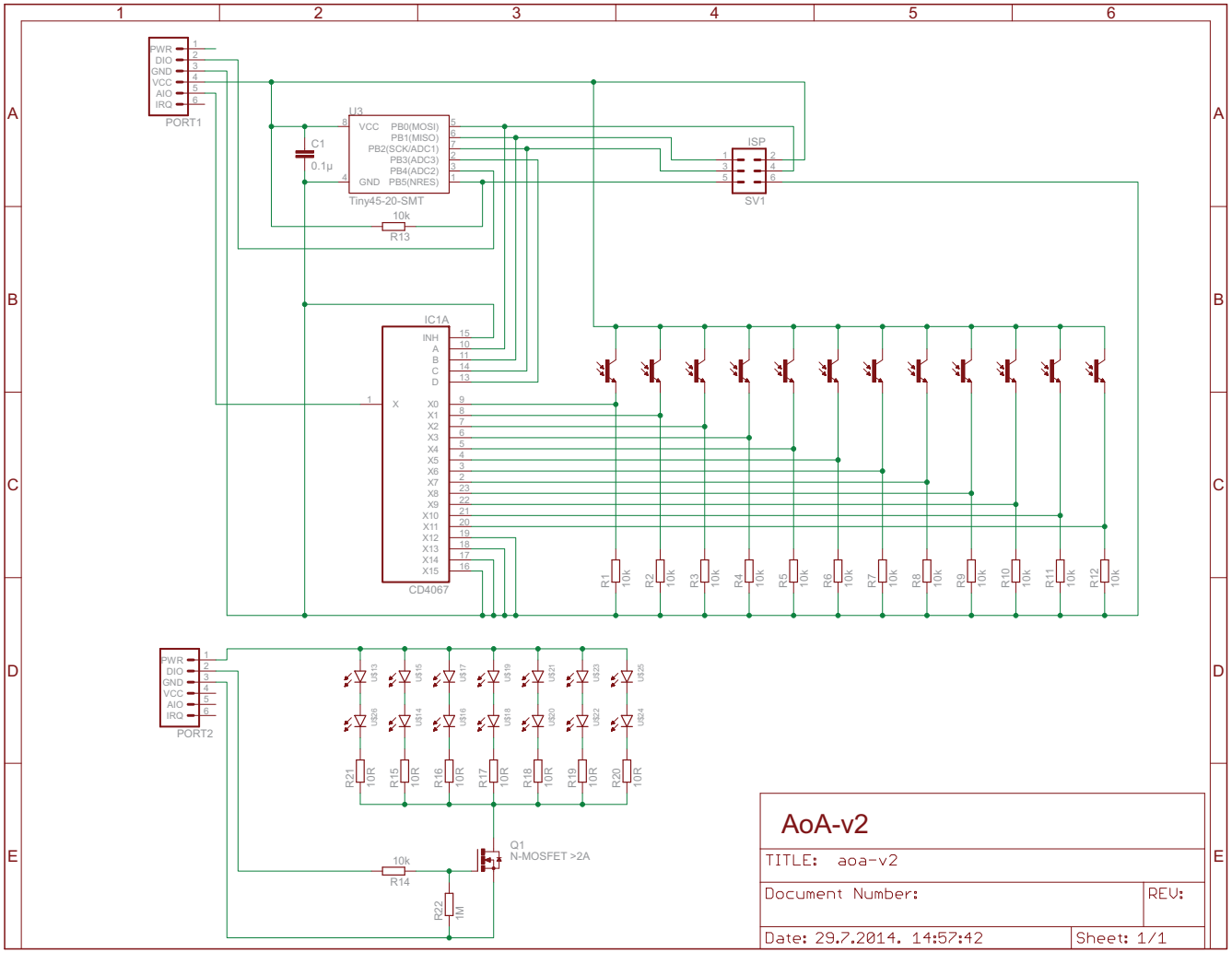


(a) Gornji sloj.



(b) Donji sloj.

Slika P.1: Raspored elemenata i vodova na pločici osjetila.



Slika P.2: Shema osjetila.

P4 Algoritam za estimaciju položaja

Algoritam 1 Pseudokod algoritma za estimaciju položaja.

S = {IDLE, TOKEN, VISITED, DONEP1, IDLESTITCH, STITCHING, WAITLOC }
S_{INIT} = {IDLE} S_{TERM} = {WAITLOC}

```
 1: IDLE
 2:     Receiving SetThr
 3:         aoa.set_threshold()
 4:
 5:     Receiving GetAoA
 6:         aoa = aoa.estimate_aoa()
 7:         if aoa != AOA_FAIL then                                  ▷ Susjed.
 8:             g_neighbors ← message.source
 9:             if !g_tree.parent then
10:                 g_tree.parent = message.source
11:             end if
12:             g_aoa ← aoa
13:             Broadcast(AoAresponse, aoa)
14:         end if
15:
16:     Receiving Token
17:         Broadcast(SetThr)
18:         aoa.led_on()
19:         delay()
20:         Broadcast(GetAoA)
21:         aoaAlarm.set()
22:         Become TOKEN
23:
24:     Receiving AoAresponse
25:         AoaMonitor()
26:
27:
28: TOKEN
29:     Receiving AoAresponse
30:         g_neighbors ← message.source
31:         if message.is_parent then
32:             g_tree.children ← message.source
33:         end if
34:         g_aoa ← aoa
35:
36:     Alarm aoaAlarm
37:         donep1 = !g_tree.children
38:         aoa.led_off()
39:         if g_tree.parent then
40:             Send(Token, donep1) to g_tree.parent
41:         else
42:             ForwardToken()
43:         end if
44:         if donep1 then
45:             Become DONEP1
46:         else
47:             Become VISITED
48:         end if
49:
50:
```

```
51: VISITED
52:     Receiving SetThr
53:         aoa.set_threshold()
54:
55:     Receiving GetAoA
56:         if message.source ∉ g_neighbors then
57:             break;
58:         end if
59:         aoa = aoa.estimate_aoa()
60:         if aoa != AOA_FAIL then                                  ▷ Susjed.
61:             g_aoa ← aoa
62:             Broadcast(AoAresponse, aoa)
63:         end if
64:
65:     Receiving Token
66:         ForwardToken()
67:
68:     Receiving AoAresponse
69:         AoaMonitor()
70:
71:
72: DONEP1
73:     Receiving SetThr
74:         aoa.set_threshold()
75:
76:     Receiving GetAoA
77:         if message.source ∉ g_neighbors then
78:             break;
79:         end if
80:         aoa = aoa.estimate_aoa()
81:         if aoa != AOA_FAIL then                                  ▷ Susjed.
82:             g_aoa ← aoa
83:             Broadcast(AoAresponse, aoa)
84:         end if
85:
86:     Receiving AoAresponse
87:         AoaMonitor()
88:
89:     Receiving DoIniLoc
90:         IniLoc()
91:         Become IDLESTITCH
92:
93:
94: IDLESTITCH
95:     Receiving StartStitch
96:         g_children_done = []
97:         if !g_tree.children then                                 ▷ List.
98:             Send(LetUsStitch, pos) to g_tree.parent
99:             Become WAITLOC
100:        else
101:            Become STITCHING
102:        end if
103:
104:
```

```
105: STITCHING
106:     Receiving LetUsStitch
107:         STITCH(message.pos) g_done_children ← message.source
108:         if g_done_children==g_tree.children then
109:             if g_tree.parent then
110:                 Send(LetUsStitch, pos) to g_tree.parent
111:                 Become WAITLOC
112:             else
113:                 Become LOCALIZED                          ▷ Algoritam je završio.
114:             end if
115:         end if
116:
117:
118: procedure AOAMONITOR
119:     if message.destination ∈ g_neighbors then
         ▷ Čvor u statusu IDLE ili TOKEN ne zna sve svoje susjede stoga mora zapisivati sva mjerenja osim
     onih u kojima čvor u statusu TOKEN nije njegov susjed.
120:         if (g_status==IDLE or g_status==TOKEN) or message.source ∈ g_neighbors then
121:             g_aoa ← message.aoa
122:         end if
123:     end if
124: end procedure
125:
126: procedure FORWARDTOKEN
127:     if message.source!=g_tree.parent and message.donep1 then
128:         g_done_children ← message.source
129:     end if
130:     if !g_tree.parent then                                ▷ Korijen.
131:         if !g_children_to_forward then
132:             g_children_to_forward = g_tree.children - g_done_children
133:             if !g_children_to_forward then                ▷ Faza 1 je završila.
134:                 Broadcast(DoIniLoc)
135:                 Broadcast(StartStitch)
136:                 INILOC()
137:                 Become STITCHING
138:             end if
139:         end if
140:     else                                                  ▷ Regularan čvor.
141:         if message.source==g_tree.parent then
142:             g_children_to_forward = g_tree.children - g_done_children
143:         end if
144:         if g_children_to_forward then
145:             child ← g_children_to_forward
146:             Send(Token) to child
147:         else
148:             Send(Token) to g_tree.parent
149:             donep1 = g_done_children==g_tree.children);
150:             if donep1 then
151:                 Become DONEP1
152:             end if
153:             Send(Token, donep1) to g_tree.parent
154:         end if
155:     end if
156: end procedure
```

Tablica P.1: Pregled statusa i tipova poruka u pojedinim fazama algoritma.

1. Faza: Prikupljanje podataka		
Status		**Opis**
IDLE		Početni status, u kojem čvorovi objavljuju svoje estimacije azimuta i osluškuju te zapisuju ista od drugih čvorova.
TOKEN		Čvor u ovom statusu koordinira mjerenjima. U danom trenutku uključuje diode čiji azimut estimiraju ostali čvorovi. Najviše jedan čvor može biti u ovom statusu.
VISITED		Čvor koji je već bio u statusu TOKEN, a sad poruku Token isključivo prosljeđuje svojim susjedima u stablu. Čvor u ovom statusu poznaje sve svoje susjede.
DONEP1		Čvor koji ili nema djece ili su mu sva djeca u statusu DONEP1. Čvor je spreman za drugu fazu odnosno u ovom statusu još samo obavlja mjerenja i prema potrebi osluškuje i zapisuje mjerenja ostalih čvorova.
Poruka	**Format**	**Opis**
SetThr	/	Čvorovi po primitku ove poruke mjere ambijentalno osvjetljenje u IR spektru
GetAoA	/	Čvorovi po primitku ove poruke mjere osvjetljenje u IR spektru te ukoliko je najveći izmjereni iznos iznad postavljenog praga estimiraju njegov azimut.
AoAresponse	`destination`[1] `aoa`[4], `is_parent`[1]	Poruka objave estimiranog azimuta, sadrži ID čvora čiji azimut se odredio, vrijednost estimiranog azimuta u lokalnom koordinatnom sustavu te zastavicu (engl. *flag*) kojom se određuje da li je čvor čiji azimut se objavljuje izabran kao roditelj u stablu.
Token	`donep1`[1]	Poruka koja obilazi stablo i služi za koordinirano obavljanje mjerenja, detekciju susjeda i kreiranje samog stabla. Sadrži zastavicu kojom se prilikom slanja poruke roditelju objavljuje da li je čvor završio s prvom fazom, odnosno da li mu i dalje treba prosljeđivati Token poruke.
2. Faza: estimacija lokacija i spajanje		
Status		**Opis**
IDLESTITCH		Nakon estimacije lokacija čvorova susjeda čvorovi iz statusa DONEP1 prelaze u ovaj status.
STITCHING		Korijen i unutarnji čvorovi stabla nalaze se u ovom statusu dok čekaju LetUsStitch poruke od svoje djece.
WAITLOC		Nakon primitka LetUsStitch poruka od sve svoje djece i slanja istog tipa poruke roditelju, čvorovi ulaze u ovaj status u kojem ostaju do eventualnog primitka svojih koordinata.
Poruka	**Format**	**Opis**
DoIniLoc	/	Poruka objave početka druge faze algoritma i čvorovima daje nalog da obave estimaciju položaja svojih susjeda.
StartStitch	/	Poruka kojom započinje proces spajanja koordinatnih sustava i na koju u *ConvergecastStitch* algoritmu reagiraju isključivo listovi.
LetUsStitch	`pos`[64]	Poruka kojom se traži od roditelja da priduži poslane lokacije svojima u procesu spajanja koordinatnih sustava.

P5 Analiza potrošnje memorije

Tablica P.2: Pregled potrošnje memorije u pojedinim fazama algoritma. Varijabla NOD označava ukupan broj čvorova mreže, NEI maksimalan broj susjeda, a MEA broj mjerenja koji može doseći NEI·(NEI+1)·2. AoA, MeasV i Pos označavaju veličine odgovarajućih struktura, redom 4, 10 i 9 byte-a. Oznake faza algoritma su 1: prva faza, 2p: druga faza priprema, 2o: orijentacija, 2a: angulacija, 2i: inicijalizacija spajanja, 2s: spajanje.

Ime strukture	**Veličina [byte]**	**Faza algoritma**					
		1	2p	2o	2a	2i	2s
`g_id`	1	✓	✓	✓	✓	✓	✓
`g_status`	1	✓	✓	✓	✓	✓	✓
`g_message`	66	✓	✓	✓	✓	✓	✓
`g_neighbors`	NOD	✓	✓	✓	✓	✓	✓
`g_num_neighbors`	1	✓	✓	✓	✓	✓	✓
`g_done_children`	NEI	✓	-	-	-	✓	✓
`g_num_done_children`	1	✓	-	-	-	✓	✓
`g_children_to_forward`	NEI	✓	-	-	-	✓	✓
`g_num_children_to_forward`	1	✓	-	-	-	-	-
`g_jbest_orientation`	4	-	-	✓	-	-	-
`g_iter_orientation`	4	-	-	✓	-	-	-
`g_jbest_angulation`	4	-	-	-	✓	-	-
`g_iter_angulation`	4	-	-	-	✓	-	-
`g_tree`	2+NEI	✓	✓	✓	✓	✓	✓
`g_aoa`	AoA· MEA	✓	✓	-	-	-	-
`g_num_aoa`	1	✓	✓	-	-	-	-
`g_fi`	4· NEI^2	-	✓	✓	-	-	-
`g_alpha_sensors`	4· NEI	-	-	✓	-	-	-
`g_alpha_sensors_tmp`	4· NEI	-	-	✓	-	-	-
`g_measurements`	MeasV· NEI^2-NEI	-	-	✓	✓	-	-
`g_neighbors_positions_tmp`	Pos· NEI	-	-	-	✓	-	-
`g_neighbors_positions`	Pos· NEI	-	-	-	✓	✓	-
`g_destination_cluster`	Pos· NOD	-	-	-	-	✓	✓
`g_num_destination`	1	-	-	-	-	✓	✓
`g_source_cluster`	Pos· NOD	-	-	-	-	-	✓
`g_num_source`	1	-	-	-	-	-	✓

Životopis

Damir Arbula rođen je 1978. godine u Zadru. Nakon završene prirodoslovno-matematičke gimnazije, 1997. godine upisuje Fakultet elektrotehnike i računarstva u Zagrebu. Diplomirao je u rujnu 2002., smjer radiokomunikacije i profesionalna elektronika, a magistrirao u polju elektrotehnike, grana automatika, u listopadu 2008. Od 2004. godine radi na Tehničkom fakultetu u Rijeci na radnom mjestu asistenta, a izvan matične institucije kao istraživač u sklopu projekta "Integrirano upravljanje robotskim sustavima u složenim okruženjima" pod vodstvom prof. dr. sc. Zdenka Kovačića. U nastavi sudjeluje na nekoliko kolegija, iz područja automatike, radiokomunikacija i računarstva na preddiplomskim studijima elektrotehnike i računarstva, te na diplomskom studiju računarstva. Glavno područje njegovog stručnog interesa su bežične mreže osjetila s posebnim naglaskom na raspodijeljene algoritme i metode za estimaciju položaja. Aktivan je i u području asistivnih tehnologija gdje s kolegama s Tehničkog fakulteta u Rijeci od 2009. godine razvija sustav za upravljanje domom pomoću govora.

Popis objavljenih djela

1. Arbula, D., Lenac, K., "Pymote: high level Python library for event based simulation and evaluation of distributed algorithms", International Journal of Distributed Sensor Networks, Vol. 2013, Localization in Wireless Sensor Network, Special Issue on, 2013, str. 52-63.
2. Martinović, A., Arbula, D., Jeričević, Ž., "Web service for separating components in the exponential decay process", Information & Communication Technology Electronics & Microelectronics, 36th International Convention on, 2013. str. 188-192.
3. Arbula, D., "Distributed Algorithm for Anchor-Free Network Localization Using Angle of Arrival", Industrial Electronics ISIE 2008. IEEE International Symposium on, 2008., str. 792-797.

Biography

Damir Arbula was born in 1978 in Zadar. Upon graduating Natural Sciences and Mathematics programme in gymnasium, in 1997 he enrolled at Faculty of Electrical Engineering and Computing, University of Zagreb. He received his dipl.ing. (BSc) and mr.sc. (MSc) degrees in electical engineering in 2002 and 2008, respectively. Since 2004 he was a teaching assistant at Faculty of Engineering, University of Rijeka, as well as junior researcher within the project "Integrated Control of Robotic Systems in Complex Environments", under the mentorship of professor Zdenko Kovačić. He has been involved in several courses within undergraduate studies of both electrical engineering and computer engineering, as well as within the graduate study of computer engineering. The main area of his professional interests are wireless sensor networks with a particular focus on distributed algorithms and localization. From 2009 he is actively involved in the field of Assistive technology developing, along with his colleagues from Technical faculty, system for home control using voice interface.