Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

MOBILITA V AD-HOC SIEŤACH

(Diplomová práca)

Bratislava, 2014

Dominika Fedáková

Katedra Informatiky

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

MOBILITA V AD-HOC SIEŤACH

(Diplomová práca)

Dominika Fedáková

Odbor: 2508 Informatika

Vedúci: prof. RNDr. Rastislav Královič, PhD.

Bratislava, 2014

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Dominika Fedáková
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Mobilita v ad-hoc sieťach

Cieľ: Cieľom práce je študovať typy mobility (mobility patterns) v ad-hoc siťach mobilných zariadení. V súčasnosti existuje viacero modelov opisujúcich pohyb mobilných zariadení v sieti. Väčšinou sú to buď jednoduché pravdepodobnostné modely založené na náhodnom pohybe, alebo ad-hoc modely s veľa parametrami vyvinuté pre konkrétnu sieť, ktoré sa dajú analyzovať iba experimentálnymi metódami. Cieľom práce v prvom rade parogramovať jednoduchú aplikáciu (napr. v prostredí android) na zber dát o polohe a s jej pomocou získať data o pohybe používateľov počas dňa. Na takto získaných dátach sa potom porovnajú rôzne modely; hlavným cieľom je zistiť, či existuje jednoducho analyzovateľný pravdepodobnostný model, ktorý by v dostatočnej miere zodpovedal reálnym dátam, prípadne navrhnúť taký model (možno ako kombináciu/rozšírenie existujúcich modelov).

Vedúci: prof. RNDr. Rastislav Královič, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 21.11.2011

Dátum schválenia: 24.11.2011

prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že diplomovú prácu som vypracovala samostatne s použitím uvedenej literatúry a pod dohľadom môjho vedúceho práce.

. . . . . .

Poďakovanie

Chcela by som poďakovať môjmu vedúcemu prof. Královičovi za pomoc, inšpiratívne rady a za dohľad nad mojou činnosťou. Ďalej by som chcela poďakovať všetkým dobrovoľníkom, ktorí boli ochotní nosiť moju aplikáciu na zbieranie GPS dát a bez ktorých by nebola možná analýza pohybu. Špeciálne poďakovanie patrí mojej rodine za ich pomoc a podporu.

Abstrakt

Práca sa zaoberá modelmi pohybu v ad-hoc sieťach mobilných zariadení. Naprogramovali sme aplikáciu pre android a nazbierali sme dáta skutočného pohybu používateľov - mobilných uzlov. Navrhli sme metodiku porovnávania úspešnosti modelov (podoby na reálne dáta). Vybrali sme typické vlastnosti pohybu, ktoré sme porovnávali v simulovaných a reálnych dátach. Z pozorovaní ostatných modelov a z analýzy vlastností reálnych dát sme navrhli vhodný model (ACRM model), ktorý je jednoduchý pre matematickú analýzu. Zároveň sa v určitej miere podobá na reálny pohyb (vlastnosti, ktoré sme sledovali).

KĽÚČOVÉ SLOVÁ: ad-hoc sieť, mobilita, android, GPS aplikácia, model pohybu, ACRM model, Random Walk, Random Waypoint, Gauss-Markov model, pravdepodobnostný Chiangov model, Levy Walk model, Akaike informačné kritérium, MLE, dvojvýberový Kolmogorov-Smirnov test

Abstract

The thesis is focusing on mobility models in ad-hoc networks composed of mobile devices. We made a new android application and collected real data tracks of mobile devices' users. We suggested a method of comparing mobility models and getting their success rate or similarity to real data tracks. We chose a typical properties for comparing simulated and real data tracks. Learning from other mobility models and from evaluation of real data properties, we developed a new mobility ACRM model that is simple enough for mathematical analysis. At the same time, it is similar to real data tracks (at least properties that we tested).

KEYWORDS: ad-hoc network, mobility, android, GPS application, mobility model, ACRM model, Random Walk, Random Waypoint, Gauss-Markov model, probabilistic Chiang's model, Levy Walk model, Akaike information criterion, MLE, two-sample Kolmogorov-Smirnov test

Obsah

Úvod

V dnešnej dobe stále sa rozširujúcich bezdrôtových zariadení, vznikajú nové výzvy vytvárania dočasných ad-hoc sietí, kde koncové zariadenia slúžia ako body prepojenia siete. Ad-hoc sieťam sa venuje veľká pozornosť a navrhujú sa nové protokoly, ktoré by čo najlepšie zohľadnili stále sa meniacu štruktúru ad-hoc sietí a optimalizovali routovacie cesty (rozhodovanie). Poslednou fázou návrhu nového protokolu je testovanie efektívnosti protokolu. Okrem teoretického dôkazu vylepšenia výkonnosti je dôležité ukázať úspešnosť protokolu v praxi. Testovanie s reálnymi zariadeniami je náročné a drahé, preto sa využívajú simulácie s použitím umelých algoritmov pre určenie pohybu mobilných uzlov (MU). Problém modelovania správania sa mobilných uzlov v sieti nemá priame a jednoznačné riešenie. Mobilnosť a strata spojenia medzi MU spôsobujú mnoho problémov pri navrhovaní vhodnej routovacej schémy pre efektívnu komunikáciu. V našej práci sa budeme venovať algoritmom pre pohyb MU (Kapitola 1).

Našim cieľom bolo vziať často využívané a novo-navrhnuté algoritmy pre pohyb MU, navzájom ich porovnať a navrhnúť najlepší model, ktorý je dostatočne jednoduchý pre matematický popis, a ktorý by zároveň čo najlepšie odzrkadľoval skutočný pohyb MU. Prvou fázou bolo naprogramovanie vlastnej aplikácie pre zbieranie GPS dát pre pohyb MU. 42 dobrovoľníkov si nainštalovalo našu aplikáciu a nosili ju na android zariadeniach, čím sme dostali informácie o ich pohybe (reálne dáta). Vytvorenie vlastnej aplikácie sa ukazovalo ako jednoduchá úloha. V skutočnosti to však zabralo mnoho času, keď sme pri testovaní aplikácie narazili na mnoho technických problémov, ktoré bolo potrebné vyriešiť pre správne zozbieranie dát (Kapitola 2).

Po získaní reálnych dát bolo dôležité navrhnúť vhodný model popisu pohybu a meto-

diku porovnávania rôznych algoritmov pre pohyb MU. Zadefinovali sme si vlastnosti pohybu, ktoré sú kľučové pre popis celej cesty a následne sme tieto vlastnosti analyzovali z pohľadu reálnych dát. Výsledky sme využili pri návrhu nového umelého algoritmu (modelu pohybu) ACRM „Aimed Constrained Random Movement“ modelu. Modely pohybu môžeme klasifikovať do dvoch základných kategórií:

- Syntetický pohyb - pohyb MU je určený náhodne, závisí len od nastavenia zopár parametrov pre náhodné distribúcie
- Sledovaný pohyb - pohyb MU je ovplyvnený vonkajšími podmienkami (pohybom celej komunity, určenými cestami pohybu, destináciami, ...)

V práci sme sa zamerali na porovnávanie syntetických modelov pohybu. Avšak náš navrhnutý ACRM model je na hrane týchto dvoch kategórií. V svojej najjednoduchšej podobe sa radí medzi (úspešnejšie) syntetické pohyby. Po miernej modifikácii, napr. po zmene výberu cieľa cesty, môžeme ACRM model zahrnúť do kategórie zložitejších sledovaných pohybov (Kapitola 3).
Posledným krokom bolo navzájom porovnať modely pohybu a reálne nazbierané dáta a tým určiť mieru úspešnosti modelu pre simuláciu MU. Znova sme definovali niekoľko vlastností pohybu, dôležitých hlavne pre efektivitu a úspešnosť protokolov ad-hoc sietí (Kapitola 4). Využili sme štatistické postupy: Akaike informačné kritérium, odhad maximálnej vierohodnosti MLE a dvojvýberový Kolmogorov-Smirnov test.
Je mnoho prác, ktoré popisujú rôzne modely pohybu s ich typickými odlišnosťami. Žiadna sa však nezamerala na vzájomné porovnanie modelov pre určenie úspešnosti alebo podoby na reálny pohyb MU. Preto sme navrhli metodiku, ktorú je možné rozšíriť, prípadne upraviť na konkrétne situácie, pridaním a porovnaním nových vlastností celkového pohybu.

Kapitola 1

Modely pohybu v ad-hoc sieťach

1.1 Ad-hoc sieť tvorená mobilnými uzlami

Mobilná ad-hoc sieť (MANET) je kolekcia bezdrôtových pohyblivých uzlov (mobilných nosičov) dynamicky vytvárajúcich dočasnú sieť bez použitia existencie sieťovej infraštruktúry alebo centralizovanej správy siete. Každý nosič reprezentuje používateľa (koncový bod) siete ako aj router preposielajúci packety po sieti. Aplikácie ad-hoc siete sa líšia od malých statických sietí po veľkoplošné vysoko dynamické mobilné siete.

Bez ohľadu na aplikáciu, ad-hoc sieť potrebuje efektívne distribuovaný algoritmus na výber sieťovej organizácie, linkového plánovania a routovania. Routovanie packetov môže byť odlišné pri rôznych routovacích protokoloch. Napríklad packet môže byť posielaný, až keď je zriadená celá cesta od zdroja k cieľu (DSR: The Dynamic Source Routing Protocol [6]). Ďalšou možnosťou môže byť packet posielaný uzlu, ktorý má najväčšiu pravdepodobnosť, že packet doručí do žiadaného cieľa. V takomto prípade a pri nepretržitej zmene topológie a susedov (uzlov, ktoré sú v dosahu môjho vysielača), nie je vždy zaručené, že sa packet do cieľa naozaj dostane (Opportunistic networking v „delay-tolerant networks“ DTN [7]). Preto sa pri rozhodovaní využívania konkrétneho protokolu sledujú hlavne tieto 3 vlastnosti:

- Miera úspešnosti „Success Rate“ - pomer úspešne doručených packetov ku cel-

kovému počtu packetov vygenerovaných v sieti.

- Cena „Cost“ - počet duplicitných packetov v sieti (veľmi dôležitý faktor, keďže mnoho protokolov často zahlcuje siete duplicitnými packetami).
- Oneskorenie „End-to-end delay“ - celkový čas trvajúci packetu pri cestovaní od zdroja k cieľu.

Pri dnešnom rozšírení bezdrôtových zariadení a záujme o výkonnejšie ad-hoc mobilné siete je stále potrebné vytvárať lepšie a lepšie routovacie protokoly.

1.2 Model pohybu „Mobility model“

Model pohybu je algoritmus, ktorý sa snaží čo najpresnejšie napodobňovať pohyb skutočných mobilných uzlov v sieti. Keďže priamočiary pohyb konštantnou rýchlosťou počas celej simulácie nezodpovedá správaniu skutočného mobilného uzla, model pohybu určuje zmenu rýchlosti, smeru a pauzy v určitých časových intervaloch. Je veľký záujem vytvoriť čo najlepší model, ktorý môžeme využiť pri vyvíjaní nových protokolov pre ad-hoc siete tvorené mobilnými uzlami v dvoch smeroch:

- Predpovedanie doby konektivity medzi dvoma uzlami - pri znalosti pohybu dvoch uzlov vieme vypočítať pravdepodobnosť, kedy a ako dlho bude medzi nimi spojenie (budú vo vzájomnej vzdialenosti menšej ako je dosah ich vysielania a prijímania signálov v sieti).
- Využitie pri simulácii a testovaní - testovanie protokolov a účinnosti siete na realistických (simulovaných) dátach je potrebné pre správne určenie vlastností daného protokolu, nielen na teoretickej báze, ale aj v praxi. Keďže je však reálne testovanie náročné a zložité, najčastejšie sa používajú umelé algoritmy na vytvorenie čo najlepšieho dojmu možného skutočného pohybu.

Modely pohybu primárne rozdeľujeme na syntetické modely a tzv. sledované „traces“ modely. Sledované modely sa riadia podľa pravidiel odsledovaných zo skutočne nazbieraných a analyzovaných dát pohybu mobilného uzla. V týchto modeloch sa často

využíva informácia o prostredí a podmienkach pohybu (veľa parametrov a nastavení, ktoré sa môžu líšiť pri rôznych využitiach ad-hoc sietí, napr. areál školy, zábavný park, centrum mesta,...). Často je pohyb jedného mobilného uzla ovplyvnený pohybom ostatných mobilných uzlov (tzv. skupinový „group mobility model“).
Syntetické modely generujú pohyb jedného mobilného uzla nezáväzne od ostatných. Využívajú náhodnosť a pokúšajú sa realisticky reprezentovať správanie mobilných uzlov bez predošlej informácie o reálnych pohyboch uzlov. Najčastejšie modely pri testovaní protokolov v ad-hoch sieťach sú syntetické modely pre ich jednoduchosť a dobrý matematický pravdepodobnostný model bez ohľadu na topológiu prostredia, kde budú ad-hoc siete využívané. Preto sa v tejto práci budeme venovať syntetickým modelom a ich zlepšeniu. Potrebujeme jednoduchý model (pre ľahké využitie), dobre matematicky popísateľný (aby sme vedeli hovoriť o pravdepodobnostiach stretu s daným uzlom), ktorý by čo najlepšie reprezentoval správanie reálneho mobilného uzla.
V tejto kapitole si popíšeme najčastejšie využívané a najnovšie navrhnuté modely pohybu mobilných uzlov. V [4] Camp, Boleng a Davids poskytujú výborný prieskum o najdôležitejších a najpopulárnejších umelých modeloch pohybu používaných vo výskumoch ad-hoc sietí, syntetických aj sledovaných modeloch.

1.3 Náhodný pohyb „Random Walk“

Tento model bol matematicky popísaný Einsteinom v roku 1926 [4]. Použil ho na popísanie mnohých nepredvídateľných javov v prírode. Základom tohto modelu sú intervaly pre zmenu uhlu pohybu $< 0, 2\pi >$ a rýchlosť pohybu $< min, max >$, z ktorých náhodným generátorom vyberá smer a rýchlosť pre daný mobilný uzol v danom momente. Vybrané náhodné parametre môžu trvať istú jednotku času t alebo trvajú jednotku prejdenej vzdialenosti d. Na konci jednotky sa náhodne vyberú nové parametre.
Je to model bez pamäte a spätnej informácie o predchádzajúcom pohybe. Tento pohyb tiež môžeme obmedziť na určitú 1D/2D/3D/... plochu. V momente, keby sme chceli prekročiť hranice plochy, vytvoríme odraz od steny s pravidlom „uhol dopadu

sa rovná uhlu odrazu“. V tomto modeli je veľmi dôležité, ako nastavíme jednotku zmeny smeru a rýchlosti. V roku 1921 Polya dokázal, že Random Walk sa vždy vracia na svoju pôvodnú pozíciu. Preto ak by sme nastavili malú jednotku vzdialenosti/času pred zmenou smeru a rýchlosti, tak dostaneme pohyb mobilného nosiča okolo svojho začiatočného bodu pohybu (všetky body vo vzájomnej relatívnej blízkosti).
Random Walk, tiež nazývaný Brownov pohyb, je veľmi využívaný model (niekedy zjednodušený aj na konštantnú rýchlosť pre všetky mobilné uzly v rámci celej simulácie), ktorý ale veľmi nezodpovedá reálnemu pohybu, keďže je nezávislý na svojom predošlom pohybe. Tento problém odstraňuje napríklad nižšie popísaný Gauss-Markov model. V tejto práci využijeme 2D pohyb Random Walk, pričom začiatočný bod vyberáme z obmedzenej plochy 5000x5000 m , ale ďalej pohyb neobmedzujeme (aby pohyb nebol veľmi koncentrovaný na stred, nevyužívame odrážanie od stien). Pohybujeme sa rýchlosťou $< 0.05, 55 >$ m/s a uhol otočenia je celé číslo z intervalu $< 0°, 359° >$. Zmenu smeru a rýchlosti vykonávame vždy po 50 metroch. Príklad z našej simulácie pohybu nájdeme na Obr. 1.1a.

1.4 Náhodný „Waypoint“ model pohybu

Spolu s Random Walk je to jeden z najvyužívanejších modelov pri simuláciach. Myšlienka za ním je veľmi jednoduchá. Ako vstupné parametre dostaneme miesto simulácie 2D mapu (súradnice x_1, y_1, x_2, y_2) a interval možnej rýchlosti $< min, max >$. Opäť začíname v náhodnom bode na mape. Následne si náhodne vyberieme cieľ (bod na mape) a rýchlosť pohybu. Budeme predpokladať, že mobilný uzol sa pohybuje rovnomerným priamočiarym pohybom, pokiaľ sa nedostane do vybraného cieľa. Po príchode bude čakať vybranú jednotku času (môže byť predom zvolená alebo náhodne zvolená zo zadaného intervalu). Opakujeme náhodný výber nového cieľa a novej rýchlosti.
Tento model má viacero problémov, ktoré spôsobujú odlišnosť umelého a skutočného pohybu v geografickom priestore. Keďže uvažujeme o ohraničenom priestore, väčšina uzlov bude koncentrovaná v strede plochy (spôsobí väčší počet susedov, ako by mohlo byť pri reálnych dátach). Taktiež je kritickým bodom zvolenie správnej veľkosti simu-

lovanej 2D plochy. Pri veľkej ploche spôsobíme dlhé cestovania v jednom smere. Teda pri obmedzenom čase a veľkej ploche sa môže stať, že počas celej simulácie pôjdeme celý čas priamočiarym rovnomerným pohybom (teda to nezodpovedá reálnemu pohybu). Okrem toho bolo ukázané, že model vykazuje úpadok rýchlosti počas dlhšieho trvania času. Počas simulácie sme si zvolili plochu 5000x5000 m, rýchlosť sme vždy vybrali z uniformnej distribúcie nad intervalom $< 0.05, 55 >$ m/s a nový cieľ sme vyberali pomocou uniformnej distribúcie zo zvolenej 2D plochy (Obr. 1.1b).

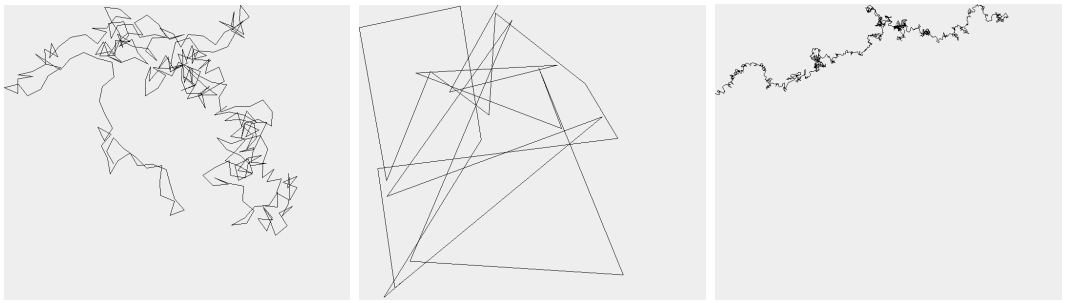
1.5 „Gauss-Markov“ model pohybu

V Gauss-Markovom modeli vieme ovplyvňovať mieru náhodností pomocou parametra vo funkcii pri rátaní nového smeru a novej rýchlosti. Čas je rozdelený na intervaly, ktoré predstavujú jednotlivé kroky, kedy sa mení smer a rýchlosť pohybu. Staré a nové premenné smeru a rýchlosti sú navzájom prepojené pomocou týchto rovníc:

$$S_n = \alpha * S_{n-1} + (1 + \alpha) * S + \sqrt{(}1 - \alpha_2) * SX_{n-1}$$
$$D_n = \alpha * S_{n-1} + (1 + \alpha) * D + \sqrt{(}1 - \alpha_2) * DX_{n-1}$$

S_n, D_n – rýchlosť a smer v časovom intervale n
α – ladiaci parameter pre náhodnosť, patrí intervalu $< 0, 1 >$
S,D - priemerná rýchlosť, smer cez všetky hodnoty n $\rightarrow \infty$
SX_{n-1},DX_{n-1} - náhodné premenné z Gaussovej distribúcie

Úplne náhodný pohyb alebo taktiež Brownov pohyb získame nastavením hodnoty $\alpha = 0$. Lineárny pohyb získame nastavením parametra $\alpha = 1$. Teda momentálny stav náhodnosti je určený meniacim sa parametrom α. V každom intervale je nasledujúce miesto pohybu rátané pomocou rovníc so základom momentálneho miesta pohybu:



(a) Random Walk (b) Random Waypoint (c) Gauss-Markov model

Obr. 1.1: *Pohyb modelov po 2D ploche part 1*

$$X_n = X_{n-1} + S_{n-1}\cos(D_{n-1})t$$

$$Y_n = Y_{n-1} + S_{n-1}\sin(D_{n-1})t$$

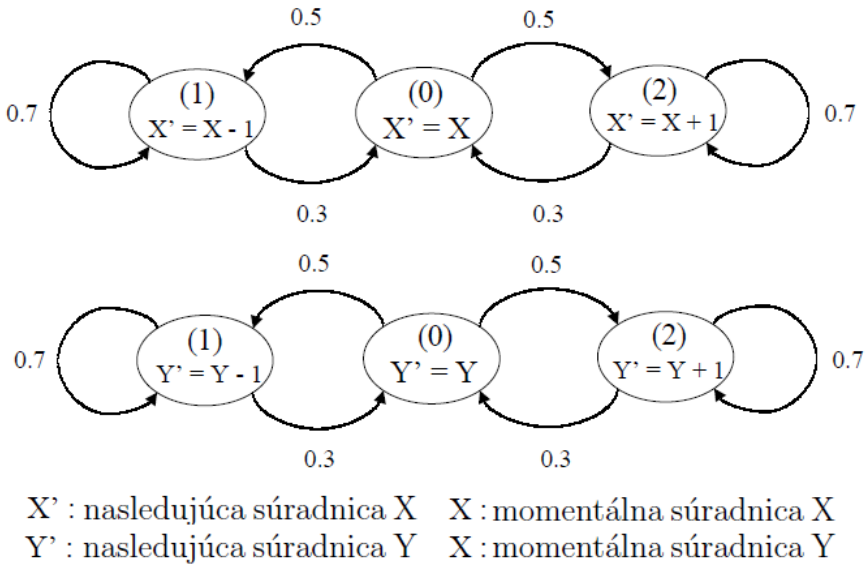
t - časový interval

Tento model zobrazuje pohyb omnoho realistickejšie ako modely Random Walk a Random Waypoint, ale nie vždy vykazuje lepšie výsledky v udržiavaní stabilných a rýchlych routovacích ciest. Týmto štúdiom sa podrobnejšie zaoberá štúdia Natarajana Meghanathana, 2010 [3]. Pri simulácii sme zvolili časový interval 2 sekundy, $\alpha = 0.45$, priemernú rýchlosť $S = 4.2865$ m/s (priemer podľa reálnych dát), smer $D = 0$, Gaussovu distribúciu pre rýchlosť so strednou hodnotou S a štandardnou odchýlkou 20 (ohraničenou do intervalu $< -20, 20 >$, pričom pri zápornom smere pôjdeme opačným smerom) a Gaussovu distribúciu pre smer so strednou hodnotou D a štandardnou odchýlkou 180 (Obr. 1.1c).

1.6 Pravdepodobnostný „Chiangov“ model

Môžeme to prirovnať pravdepodobnostnej verzii Random Walk. C. Chiang chcel do simulácie pohybu uzlov pri testovaní svojho navrhnutého multicast protokolu zapracovať pozorovanie z reálneho života: „Namiesto náhodného modelu pohybu skúmame pravdepodobnostný model, ktorý poskytuje stabilnejší pohyb uzla.“ Do-

voľoval otočenie len v 4 smeroch (sever, západ, juh, východ), avšak musí sa stále hýbať, nemá čas určený na pauzy. Navyše pravdepodobnosť, že sa mobilný uzol bude hýbať v rovnakom smere, je väčšia ako pravdepodobnosť, že uzol zmení smer pohybu. Správne definované hodnoty prikazujú, aby sme pri prechode od minulej pozície do nasledujúcej pozície prešli cez momentálnu predrátanú pozíciu. Model Chiang realizoval homogénnym trojstavovým Markovovým reťazcom[1] pre súradnicu X a rovnakým Markovovým reťazcom pre súradnicu Y Obr. 1.2.



Obr. 1.2: *Vývojový diagram pre pravdepodobnostný Chiangov model.[8]*

Pravdepodobnosti prechodu medzi stavmi v jednom kroku môžeme vyjadriť pravdepodobnostnou maticou P, kde prvok p_{ij} predstavuje pravdepodobnosť prechodu zo

[1]Markovov reťazec je postupnosť náhodných premenných $X_0, X_1, X_2...$, taká že $Pr(X_t|X_0, ..., X_{t-1}) = Pr(X_t|X_{t-1})$,teda hodnota X v čase t závisí len od hodnoty X v čase $t-1$ a nie ďalších predchádzajúcich hodnôt. Pri homogénnych Markovových reťazcoch hodnota $Pr(X_t|X_{t-1})$ nezávisí od t. Trojstavový znamená, že hodnota X_t môže nadobúdať tri hodnoty, podľa toho v akom je stave[9].

stavu i do stavu j, v našom prípade maticou:

$$P = \begin{pmatrix} 0 & 0.5 & 0.5 \\ 0.3 & 0.7 & 0 \\ 0.3 & 0 & 0.7 \end{pmatrix}$$

Tento model je viac pravdepodobnostný ako náhodný, a preto v niektorých prípadoch vykazuje lepšie výsledky[8].

Pri našej simulácii sme využili hore popísanú pravdepodobnostnú maticu pre súradnice X a Y. Celú simuláciu sa pohybujeme rovnakou rýchlosťou 4.2865 m/s (priemer podľa reálnych dát) a dĺžka jedného kroku po súradnici X aj Y je 10 metrov (Obr. 1.3a).

1.7 „Levy Walk“ model pohybu

Je to model, ktorý bol navrhnutý po analýze dát, získaných pomocou reálnych ciest ľudí (mobilných uzlov). Mohli by sme ho preto zaradiť medzi sledované modely, ale keďže tento model využíva náhodné distribúcie, stále ho radíme medzi syntetické modely.

V práci Brockman et al.[10], ktorý sledoval pohyb ľudí pomocou pohybu na bankových účtoch (teda na úrovni tisíciek kilometrov), je ukázaný Levy-Walk model pohybu ľudí. Taktiež Gonzales et al.[11] ukázal, že jednotlivé cesty (vzdialenosti medzi dvoma bodmi pred zmenou smeru) nasledujú „heavy-tailed“ distribúciu[2]. Použili informácie o pohybe od 100 000 ľudí prostredníctvom záznamov umiestnenia prenosovej telefónnej stanice, na ktorú boli pripojení, keď prebiehal hovor alebo iná aktivita. V oboch prípadoch však máme veľmi abstraktný pohľad na pohyb, keďže je zachytávaný zriedkavo a na úrovni kilometrov.

Neskôr však Rhee et al.[12] sledoval pohyb pomocou GPS trackov na úrovni metrov každých 30 sekúnd (ukázal heavy-tailed vlastnosť dĺžky ciest a teda, že táto vlastnosť sa zachováva v krátkom aj dlhom období, na úrovni cestovania metrov aj kilometrov).

[2]heavy-tailed distribúcia má ešte ťažší chvost ako exponenciálna funkcia, teda pri pravom chvoste je veľmi veľká pravdepodobnosť výberu nízkych čísel a pravdepodobnosť výberu vysokých čísel klesá rýchlejšie ako exponenciálna funkcia

Analýzou nazbieraných dát vybral najlepšie sa hodiacu distribúciu pre dĺžky jednotlivých ciest, dĺžky pauz, rýchlosti a zmenu smeru s príslušnými najlepšími parametrami pre dané distribúcie. Model funguje podobne ako Random Walk, len pracujeme so špeciálnymi distribúciami pri výbere hodnôt (Obr. 1.3b).

- Dĺžka jednotlivých ciest - nasleduje obmedzenú Pareto distribúciu[3], pre $\alpha = 2$ je táto distribúcia redukovaná na Brownov pohyb. (V našej simulácii využijeme $\alpha = 0.9020$, pričom ju zhora obmedzíme hodnotou 3000 m a zdola 23 m.)

- Dĺžka pauz - nasleduje obmedzenú Pareto distribúciu. (V našej simulácii sme pracovali bez využitia pauz.)

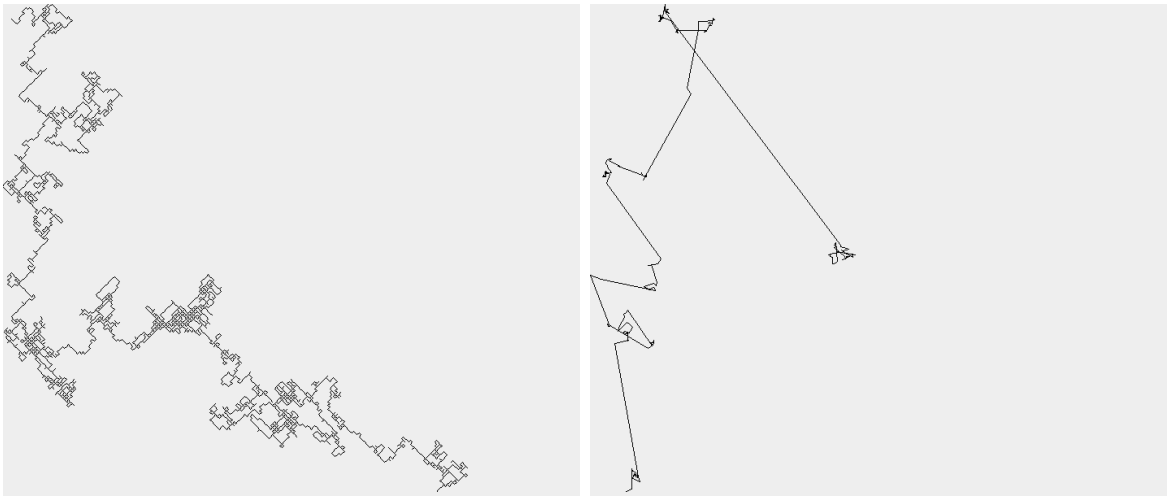
- Rýchlosť - podľa pozorovania, že pri dlhších trasách ľudia využívajú dopravný prostriedok a majú teda vyššiu rýchlosť ako pri krátkych trasách, vyvinuli špeciálny vzorec na rátanie rýchlosti podmienený dĺžkou momentálnej cesty: $\Delta t_f = kl^{(1-\rho)}, 0 \leq \rho \leq 1$, kde Δt_f je dĺžka trvania cesty l, k a ρ sú konštanty. Ak $\rho = 0$, čas cesty a dĺžka cesty sú priamo úmerné. V opačnom krajnom prípade ak $\rho = 1$, čas cesty je konštanta a rýchlosť cesty je priamo úmerná dĺžke cesty. Ich nazbieraným dátam najlepšie pasovali hodnoty parametrov $k = 30.55$ a $\rho = 0.89$ pre $l < 500m$ a $k = 0.76$ a $\rho = 0.28$ pre $l \geq 500m$. (Tieto prarametre využijeme aj v našej distribúcii).

- Zmena smeru - uniformná distribúcia.

V tomto modeli tiež ukázali platnosť niektorých ďalších vlastností. Je škálovateľný na krátke a dlhé plochy pohybu (teda nezáleží, či budeme chcieť pracovať na metrové alebo kilometrové vzdialenosti). Pre MSD[4] pri Brownovom pohybe je priamo úmerný t, čo nazývame normálny rozptyl v priestore (napr. 2D ploche). Pri našom obmedzenom Levy Walk modeli na začiatku sledujeme vysoký rozptyl priamoúmerný $t^\gamma, \gamma > 1$, neskôr normálny rozptyl až nízky rozptyl priamo úmerný $t^\gamma, \gamma < 1$. Pri

[3] $\frac{\alpha a^\alpha}{x^{\alpha+1}1-\left(\frac{a}{b}\right)^\alpha}$

[4] displacement alebo posunutie je v čase t priemerná vzdialenosť chodca od jeho začínajúcej pozície, pozície v čase 0. MSD („mean square displacement“) je druhá mocnina priemerného posunutia náhodného chodca



(a) Pravdepodobnostný Chiangov model (b) Levy Walk model

Obr. 1.3: *Pohyb modelov po 2D ploche part 2*

vysokom rozptyle je skrátený čas spojenia medzi dvoma uzlami, čo výrazne ovplyvní výsledky routovacích protokolov v ad-hoc sieťach. Priemerný rozptyl Levy Walk modelu je niekde medzi Random Walk (normálny až nízky rozptyl pri veľkých pauzách) a Random Waypoint (vysoký rozptyl). Preto na rozdiel od Levy Walk modelu často používaný Random Waypoint model väčšinou nadhodnocuje pri simuláciach účinnosť protokolov v oportunistických sieťach DTN[5] a podhodnocuje účinnosť protokolov v mobilných ad-hoc sieťach (MANET[6]).

[5]„delay-tolerant networks“-nemusí existovať end-to-end spojenie pri posielaní packetov, uzly sa môžu svojvôľne pripájať a odpájať (napr. pri strate signálu spojenia pri veľkej vzdialenosti)

[6]vyžaduje end-to-end spojenie pred posielaním packetov v sieti

Kapitola 2

Získavanie dát

Pri porovnávaní podoby modelu pohybu na reálny pohyb mobilných uzlov vzniká potreba zozbierania si reálnych dát na porovnávanie. Vždy je otázkou, na aké vlastnosti sa pozeráme, odkiaľ a ako sme dáta nazbierali, a preto rôzne výskumy môžu preukázať rôzne výsledky. Napríklad v [12] ukázali, že pre dáta zozbierané od dobrovoľníkov počas veľtrhu platí blízkosť distribúcie dĺžok jednotlivých ciest ku „short-tailed“ exponenciálnej funkcii, pričom v ostatných prípadoch (centrum New Yorku, DisneyLand, školská pôda) sa dĺžka jednotlivých ciest výrazne odlišuje od „short-tailed“ distribúcií. Je to spôsobené hlavne veľkým obmedzením pohybu pri veľtrhu.
Dáta sme zbierali pomocou aplikácie na android, ktorú si nainštalovalo 42 dobrovoľníkov a zbierali GPS dáta počas svojich každodenných ciest. Pozerali sme sa na ich pohyb na úrovni metrov a úrovni sekúnd pre zmenu lokácie. Jeden „track“ sme vnímali ako presun medzi dvoma miestami alebo budovami (napr. presun z domu do roboty, z internátov do školy,...). Tieto zozbierané dáta sme neskôr upravili zo „surového“ pôvodného do „upraveného“ použiteľného tvaru, kde sme zjemnili krivky pohybu (viac v sekcii 2.2 Úprava dát).

2.1 Android aplikácia na zbieranie GPS dát

Pre veľké rozšírenie smartfónov s androidom v našom okolí (oproti Apple iphone alebo Nokia Windows phone) sme sa rozhodli pri zbieraní dát vyžívať aplikáciu na android, ktorá sa bude stále dotazovať na polohu telefónu pomocou GPS a dáta si ukladať do súboru. Ak následne zistí pripojenie na internetovú sieť, tak dáta z tohto súboru nám pošle na server, aby sme s nimi mohli priebežne pracovať. Chceli sme, aby aplikácia pracovala sama na pozadí, aby s ňou mal dobrovoľník čo najmenšie problémy, vedel si to sám nainštalovať, nemusel nič nastavovať a dáta boli konzistentné. Preto sme si vytvorili vlastnú aplikáciu (v programe Eclipse s ADT pluginom), ktorá robí presne to čo potrebujeme a nič navyše, aby čo najmenej míňala batériu telefónu, a teda aby čo najmenej vadila dobrovoľníkom pri jej nosení v telefóne. Najprv to vyzeralo ako ľahký cieľ, neskôr sa však ukázalo, že tam je mnoho technických detailov, ktoré je potrebné vyriešiť. V celej kapitole budeme používať anglické výrazy pre konzistenciu názvov.

2.1.1 Spôsob zbierania GPS dát pomocou android aplikácie

Najprv sme vytvorili aplikáciu, ktorá sa pozerala na polohu telefónu pomocou GPS, ale taktiež pomocou internetového pripojenia (network), aby sme mohli sledovať pohyb mobilného uzla aj vo vnútri budovy a mohli sme robiť celodenné „tracky“. Na GPS polohu sme sa dotazovali každých 15 sekúnd z pohľadu GPS aj network providera lokácie (pomocou triedy AlarmManager [13]). Každý bod sme si zapísali do interného súboru a každých 15 minút sme vyskúšali internetové pripojenie, aby sme dáta zo súboru poslali na server. Pri testovaní však nastali problémy. Vznikali nám veľké súbory, pričom väčšinu dňa sa lokácia nemenila. 15 sekúnd sa ukázalo ako príliš dlhý interval medzi jednotlivými dotazmi, pretože sa stávalo, že sa GPS provider vypol a vždy mu trvalo dlho kým sa znova nainicializoval. Preto sme tento interval postupne zmenšovali a upravovali. Nakoniec sme sa rozhodli pre riešenie, kde sme sa dotazovali na GPS každú sekundu. Každý získaný bod lokácie sme si zapísali najprv do poľa a po získaní 10 bodov sme vyrátali priemer[1] z týchto 10 bodov, ktorý sme si

[1] neskôr sa ukázalo rátanie mediánu lepšie ako priemeru

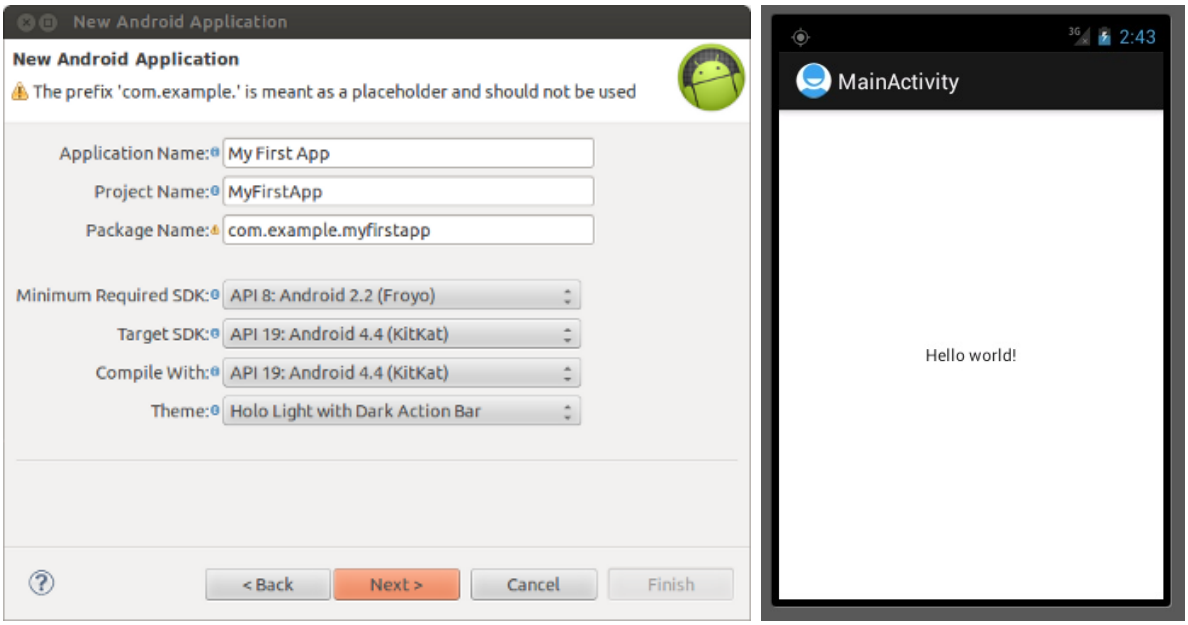
zapásali do interného súboru. Tu však nastávali ďalšie problémy. Museli sme ošetriť prípady, keď nám provider nenašiel lokáciu a vrátil nám hodnotu $(0, 0)$ alebo chybný bod - outliner (ďaleko od nášho skutočného umiestnenia, na ktorý môžeme prísť minimálne pozeraním sa na rýchlosť pri premiestňovaní). Takéto body sme nebrali do úvahy pri rátaní priemeru, ktorý zapíšeme. Prepočtom a častým dotazovaním sa výrazne zvýšila spotreba energie a našu alpikáciu bežiacu na pozadí systém zhadzoval pri nedostatku prostriedkov pre iné aplikácie. Okrem toho nám pri spustenej aplikácii nevydržala batéria telefónu ani celý deň (výrazny nedostatok, pre ktorý dobrovoľníci aplikáciu vypínali).

Ďalším problémom sa ukázalo získanie lokácie pomocou network providera. Ak sme boli pripojený na wi-fi sieť, stávalo sa, že sa menil access point, na ktorý sme napojený. V tom prípade sa ale zmenila z pohľadu providera naša lokácia, aj keď sme v skutočnosti telefónom nehýbali. Preto nám vznikalo veľké skreslenie pohybu - veľa malých ciest v jednom okolí, aj keď to v skutočnosti malo byť dlhé čakanie na jednom mieste (napr. 8 hodín v práci je zväčša čakanie na jednom mieste, pričom nám sa to ukazovalo ako presúvanie sa v rámci 100 metrov pri nepresnom určení lokácie od wi-fi network providera).

Z týchto dôvodov sme sa rozhodli aplikáciu prepracovať nanovo. Dôraz sme dali na presun mobilného uzla medzi budovami a dotazovali sme sa iba na GPS provider. Zrušili sme AlarmManager a miesto neho sme využili triedu LocationManager, od ktorého sme dostali informáciu vždy pri zmene lokácie (výrazne sa zmenšila veľkosť súboru). Pri vypnutom GPS sa naša aplikácia stávala nečinnou (čakala na zapnutie GPS, prípadne pripojenie na internet pre poslanie súboru na server). Technické detaily výsledného riešenia popisujeme nižšie.

2.1.2 Vytvorenie android aplikácie

Prvým krokom je vytvorenie aplikácie [16]. Využitie prostredia Eclipse s ADT pluginom nám výrazne uľahčí prácu pre vytvorenie našej prvej aplikácie pre android. Dôležité parametre pri vytváraní aplikácie, ktoré by som chcela spomenúť, sú Minimum Required SDK, Target SDK a Compile With. Minimum Required SDK je najnižšia verzia androidu, ktorú táto aplikácia podporuje. Je najlepšie nastaviť to



(a) (b)

Obr. 2.1: *Vytváranie novej android aplikácie v ADT prostredí [15][16]*

na čo najnižšiu verziu, aby sme pokryli veľké množstvo zariadení. Ak využívame funkcionalitu, ktorá je dostupná len na vyšších verziach androidu, ale hlavná časť aplikácie je prístupná pre všetky verzie androidu, môžeme túto funkcionalitu povoliť len pri vyšších verziach a inak bude aplikácia ponúkať hlavnú časť. Target SDK určuje najvyššiu verziu androidu, pre ktorú sme testovali a vyvíjali túto aplikáciu. Compile With je verzia platformy, na ktorú kompilujeme našu aplikáciu. Mala by to byť najvyššia dostupná verzia androidu, aby sme povolili všetky nové funkcie androidu a tým zlepšili zážitok z našej aplikácie. Package Name, ktoré si zvolíme musí byť jedinečný pre všetky balíky nainštalované v systéme android nášho zariadenia (Obr. 2.1b). Ďalej si môžeme nakonfigurovať náš projekt a vytvoriť vlastnú ikonku aplikácie. Pre začiatok sme si vytvorili prázdnu aplikáciu podľa templatu BlankActivity (Obr. 2.1b).

2.1.3 Service a jeho životný cyklus

Service je vybavenie aplikácie, ako povedať zariadeniu o niečom, čo chceme aby robil na pozadí (aj v momente, keď užívateľ aktívne neinteraguje s aplikáciou). To vedie k volaniam `Context.startService()`, čím žiada zapojiť service medzi spustené procesy a ostane spustený, kým sa samotný service nerozhodne skončiť alebo ho systém nútene zastaví z prevádzkových dôvodov (môžeme tiež používať príkaz `Context.bindService()`, ak chce aplikácia udržiavať komunikáciu so servicom). Po jednom z týchto príkazov systém vytvorí inštanciu servicu a zavolá jeho `onCreate()` metódu[2]. Následne sa zavolá metóda `onStartCommand(Intent, int, int)` s parametrami určenými klientom (aplikáciou). Service bude od tohto momentu bežať, kým sa nezavolá príkaz `Context.stopService()` alebo `stopSelf()`. V našom prípade chceme, aby GPS aplikácia bežala neustále na pozadí zariadenia, preto musíme hlavnú funkcionalitu presunúť do servicu.

Service môže bežať v dvoch verziach, podľa toho, akú hodnotu vracia pri metóde `onStartCommand()`. `START_STICKY` sa používa, ak chceme explicitne ovládať zapínanie a vypínanie servicu (teda, aj keď je náš service dlho neaktívny a systém service zhodí kvôli uvoľneniu prostriedkov a pamäte, alebo aj keď zatvoríme aplikáciu, ktorá service vytvorila, tak náš service znova požaduje spustenie). Druhou možnosťou je `START_not_STICKY` alebo `START_REDELIVER_INTENT`, ktorá sa využíva pre service, ktorý má bežať len počas vykonávania príkazov. Service ale nebude zhodený, ak má na seba naviazaných klientov pomocou `Context.bindService()`. V našom prípade, keďže chceme aby GPS aplikácia bežala na pozadí nepretržite, kým sa ju užívateľ nerozhodne vypnúť, budeme používať pri metóde `onStartCommand()` návratovú hodnotu `retutn START_STICKY`. Na toto si treba dávať pozor, keďže štandarne je nastavená návratová hodnota na `START_not_STICKY`. Až po testovaní našej aplikácie sme prišli na túto vlastnosť, ktorú je dôležité zmeniť.

Android systém sa snaží náš service (proces spustený servicom) udržať nažive, kým beží alebo je naňho naviazaný klient. Ale ak má málo pamäte a musí sa rozhodnúť, ktorý proces je najlepšie zabiť, rozhoduje sa podľa týchto pravidiel:

[2]ak inštancia servicu už existuje, metóda `onCreate()` sa nezavolá

- ak service práve vykonáva metódu `onCreate()`,`onStartCommand()` alebo `onDestroy()`, potom sa proces spustený servicom dostane do popredia, aby sa zaručilo, že sa tento kód vykoná bez násilného zabitia.
- ak bol service spustený, potom je jeho proces menej dôležitý ako procesy viditeľné pre užívateľa na obrazovke, ale je dôležitejší ako všetky ostatné procesy, ktoré nie sú viditeľné.
- ak je service naviazaný na klientov, potom nie je nikdy menej dôležitý ako najdôležitejší klient. Teda ak je jeden z jeho klientov viditeľný, potom sa považuje service za viditeľný.
- spustený service môže využiť `startForeground(int, Notification)` API, aby dostal svoj service do popredia. Zabitý môže byť len vo veľmi kritickej situácii (v praxi by sa to nemalo stávať často).

Viac o servicoch si môžete prečítať v [14].

2.1.4 Funkcionalita za GPS aplikáciou

Keďže nám nezáleží na vzhľade a jediné čo potrebujeme sú tri tlačidlá s funkcionalitou za nimi, tak sa nebudeme zaoberať užívateľským prostredím a layoutom (jediné čo spravíme je, že tam pridáme tlačidlá a text s popisom aplikácie). V súbore `res > layouts > activity_main.xml` pridáme elementy <`Button`> a elementy <`Textview`>, napríklad [3]:

```
<Button
      android:id="@+id/btnStart"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true"
      android:layout_marginBottom="23dp"
      android:text="@string/start_track"
      android:onClick="start_gpsService" />
```

[3] v premenných `@string/comment` a `@string/start_track` musíme mať nastavené hodnoty, ktoré chceme aby sa zobrazili (prípadne tam miesto premenných môžme dať priamo text, ale neodporúča sa to)

```
<TextView
       android:id="@+id/textView3"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_centerHorizontal="true"
       android:layout_alignParentBottom="true"
       android:layout_marginBottom="10dp"
       android:text="@string/comment"
       tools:context=".MainActivity" />
```

Takto sme pre tlačidlo `btnStart` nastavili pri kliknutí spustenie funkcie `start_gps-Service`, ktorá spúšťa service zbierajúci GPS dáta a riadi posielanie dát cez internetovú sieť. Ďalšie tlačidlo, ktoré sme si vytvorili v našej aplikácii je `btnStop`, pre zastavenie servicu a tlačidlo `show_settings`, kde sa vieme dostať do nastavení GPS (Obr. 2.2).

Metóda za tlačidlom `show_settings` najprv upozorní užívateľa, či sa naozaj chce premiestniť do nastavení pre zapnutie GPS a následne po potvrdení ho premiestni do nastavní, sekcie GPS:

```
public void showSettingsAlert(){
       AlertDialog.Builder alertDialog = new AlertDialog.Builder(GPS_tracker.this);
       alertDialog.setTitle("GPS settings");// Nastavenie nazvu vyskakovacieho okna
       alertDialog.setMessage("GPS is not enabled. Do you want to go to settings menu?");// Nastavenie spravy vyskakovacieho okna
       // po kliknuti OK tlacidla
       alertDialog.setPositiveButton("Settings", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog,int which) {
               Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
               GPS_tracker.this.startActivity(intent);
           }
       });
       // po kliknuti cancel tlacidla
       alertDialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int which) {
           dialog.cancel();
           }
       });
       alertDialog.show();// ukaz spravu
   }
```

Na lokalizovanie polohy telefónu používame GPS súradnice. Na to využijeme triedu LocationManager [17], ktorá poskytuje prístup k systémovým lokalizačným servicom. Využijeme tiež triedu LocationListener [22], ktorá zbiera notifíkacie od LocationManager. Takto dokážeme získať pravidelnú aktualizáciu geografickej polohy zariadenia. LocationListener musíme registrovať k príslušnému LocationManager metódou `requestLocationUpdates(string, long, float, LocationListener)`. Zadávame

tam parametre `LOCATION_INTERVAL` (typ zbierania geografickej polohy, napr. `GPS_PROVIDER, NETWORK_PROVIDER`, ...) , `LOCATION_INTERVAL` (minimálny interval času medzi určením novej polohy) a `LOCATION_DISTANCE` (minimálny interval vzdialenosti medzi určením novej polohy). Pred nastavením hodnôt `LOCATION_DISTANCE` a `LOCATION_INTERVAL` sme strávili veľa času testovaním. Pri nastavení malých hodnôt dochádza k častému dopytovaniu sa na polohu a veľkej spotrebe batérie. Pri nastavení vyšších hodnôt sme nedostávali dostatočne detailné dáta pre našu analýzu. Vždy je dobré uvedomiť si ako jemné dáta potrebujeme a možno si vyskúšať niekoľko variantov hodnôt pred konečným uvedením („releasom“) aplikácie. V našej aplikácii sme nakoniec vybrali hodnotu 5m pre `LOCATION_DISTANCE` a 2s pre `LOCATION_INTERVAL`.
Spustíme sledovanie GPS polohy a získavanie pravidelných notifikácii pri zmene polohy (metóda `onLocationChanged(Location location)` sa spustí vždy pri zaregistrovanej zmene polohy). Funkcie, ktoré sa majú vykonať po zapnutí alebo vypnutí GPS (resp. iného poskytovateľa polohy) môžeme spravovať metódami `onProviderEnabled(String provider)` a `onProviderDisabled (String provider)`. Funkcionalitu týchto metód si môžeme prepísať v triede LocationListener. Ak chceme vypnúť získavanie notifikácii pre LocationListener, musíme zavolať metódu `removeUpdates(LocationListener)`, najlepšie pri zastavení servicu v metóde `onDestroy()`.
V súbore `AndroidManifest.xml` musíme nastaviť povolenia získavať presnú geografickú polohu:

```
//AndroidManifest.xml
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />


//GPStracker.java service
int LOCATION_INTERVAL = 2000;
float LOCATION_DISTANCE = 5;
private class LocationListener implements android.location.LocationListener{
       Location mLastLocation;
       public LocationListener(String provider)
       {
          mLastLocation = new Location(provider);
       }
       public void onLocationChanged(Location location)
       {
          mLastLocation.set(location);
          //...mozeme vyuzit novu lokaciu mLastLocation.getTime(), mLastLocation.getLatitude(), mLastLocation.getLongitude()
       }
       public void onProviderDisabled(String provider)
       {
             releaseWakeLock();
                                 //...
       }
       public void onProviderEnabled(String provider)
       {
```

```
            if (mLocationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)){
                  acquireWakeLock();
                                    //...
            }
      }
      public void onStatusChanged(String provider, int status, Bundle extras)
      {
            //...
      }
}
LocationListener mLocationListener= new LocationListener(LocationManager.GPS_PROVIDER);
mLocationManager = (LocationManager) getApplicationContext().getSystemService(Context.LOCATION_SERVICE);//inicializacia LocationManager
mLocationManager.requestLocationUpdates(
                LocationManager.GPS_PROVIDER, LOCATION_INTERVAL, LOCATION_DISTANCE,
                mLocationListener); //zaciname posielat requesty na zistenie zmeny polohy

//...aktivne sledovanie GPS polohy zariadenia

mLocationManager.removeUpdates(mLocationListener); //vypnutie posielania requestov na zistenie polohy
```

Na posielanie dát cez internet používame triedu BroadcastReceiver [18] a ConnectivityManager [19]. BroadcastReceiver prijíma intent správy brodcastované v systéme. ConnectivityManager nám odpovedá na otázky ohľadom stavu pripojenia na internetovú sieť. Taktiež nás upozorní pri zmene stavu pripojenia. Pre využívanie týchto tried musíme nastaviť príslušné povolenia v `AndroidManifest.xml`:

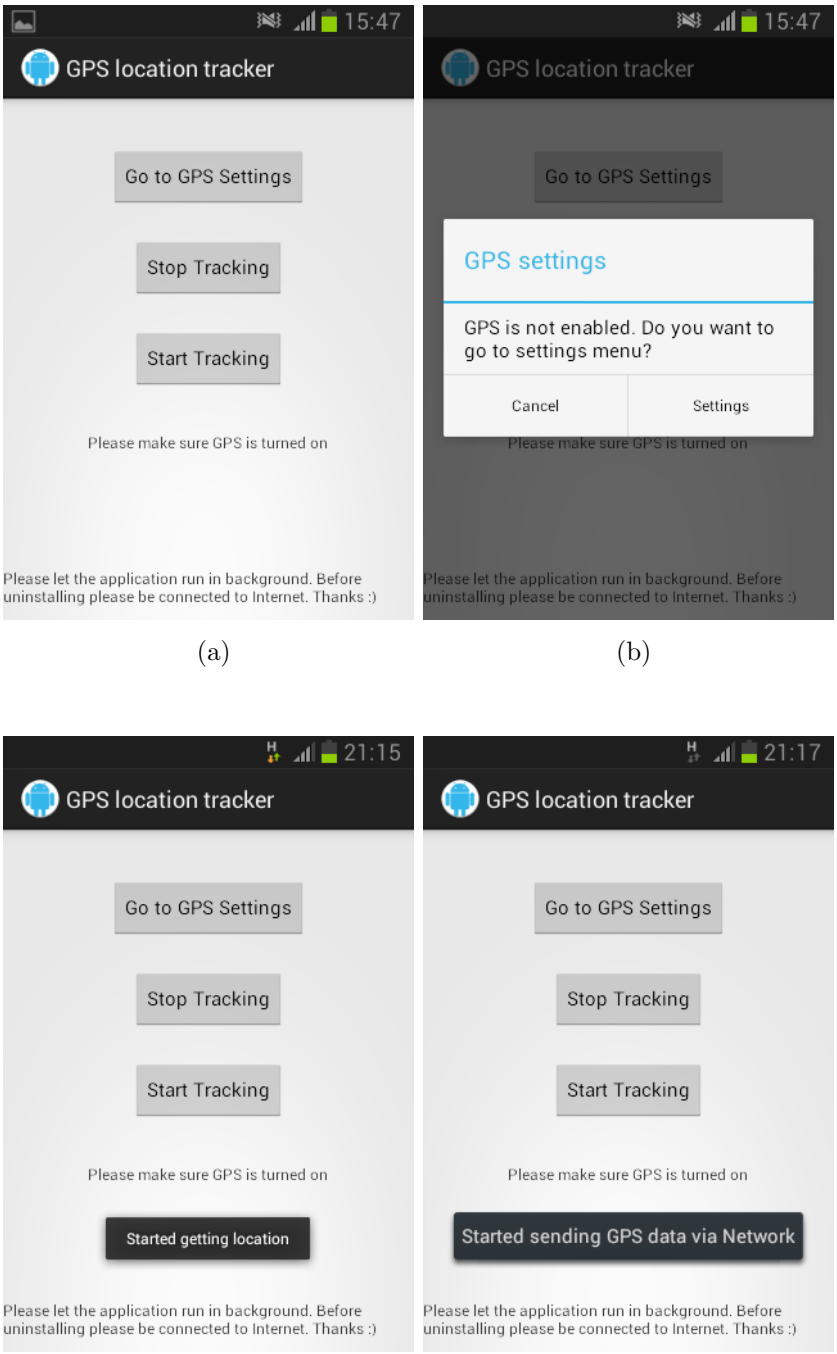
```
//AndroidManifest.xml
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />


//GPStracker.java service
BroadcastReceiver networkStateReceiver = new BroadcastReceiver() {
      //@Override
  public void onReceive(Context context, Intent intent) {
      ConnectivityManager connectivityManager=(ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
    if (activeNetworkInfo != null)
        ...
    }
  };
```

Nazbierané dáta posielame vždy, keď sa užívateľ pripojí na internet (pri minimálnej veľkosti súboru 3kB alebo vypnutí GPS). Na toto sme využili externú knižnicu Java Secure Channel od spoločnosti JCraft[23], ktorá implementuje bezpečné pripojenie a posielanie súborov na vzdialený server pomocou protokolu SSH2 (je dôležité si importovať externé knižnice JSch a JZlib[4][23]). Ak by sme volanie na server implementovali v hlavnom UI vlákne, užívateľovi by sa aplikácia javila ako zamrznutá, kým vola-

[4]knižnica podporujúca kompresiu packetov pre SSH systémy

nie na server neskončí. Android sa chce vyhnúť takejto zlej užívateľskej skúsenosti, a preto zabraňuje tejto implementácii vyhodením výnimky pri kompilácii. Z tohto dôvodu sme v našej aplikácii implementovali pripájanie sa na server v AsyncTask [24](vedľajšom vlákne, ktoré beží na pozadí). AsyncTask je pomocná trieda okolo vlákien a Handlera. Mala by sa využívať iba na rýchle operácie (ideálne trvajúce pár sekúnd). Dôležité metódy tejto triedy, ktoré je dôležité prepísať podľa potreby, sú `doInBackground(String... params)` (definuje hlavnú vykonávanú funkcionalitu), `onPreExecute()` (príkazy vykonávané pred spustením hlavnej časti), `onProgressUpdate(Void... values)` (napr. by sme mohli zadefinovať progess bar) a `onPostExecute(String result)` (príkazy vykonané po hlavnej časti). V našom prípade, pred pripojením na server upozorníme užívateľa správou o posielní dát na server a na konci správou o dokončení posielania dát:

```
public class send_file extends AsyncTask<String, Void, String> {
       @Override
       protected String doInBackground(String... params) {
              try{
                     String serverUrl = "***.sk";
                     String userName = "**********";
                     String password ="**********";
                     JSch jsch = new JSch();
                     Properties config = new Properties();
                     config.put("StrictHostKeyChecking", "no");
                     config.put("compression.s2c", "zlib,none");
                     config.put("compression.c2s", "zlib,none");
                     Session session = jsch.getSession(userName, serverUrl);
                     session.setConfig(config);
                     session.setPort(22);
                     session.setPassword(password);
                     session.connect();
                     ChannelSftp channel = (ChannelSftp) session.openChannel("sftp");
                     channel.connect();
                     File log_file = new File(getFilesDir(),filename); //nas lokalny subor s datami
              channel.put(new FileInputStream(log_file), filename,APPEND); //data z nasho lokalneho suboru pripojime na subor na
                    serveri
                                   channel.disconnect();
                     session.disconnect();
                }
              } catch(Exception e){
                     System.out.println("Sending interrupted: "+e.toString());
                     return "Sending interrupted: "+e.toString();
              }
       }
       @Override
       protected void onPostExecute(String result) {
          Toast.makeText(GPSTracker.this, "Ended sending GPS data via Network", Toast.LENGTH_SHORT).show();}
       @Override
       protected void onPreExecute() {
          Toast.makeText(GPSTracker.this, "Started sending GPS data via Network", Toast.LENGTH_SHORT).show();}
       @Override
       protected void onProgressUpdate(Void... values) {}
    }
```



Obr. 2.2: *Android aplikácia pri jej behu (a)po spustení aplikácie, (b)pri prechode do GPS settings* `Show Settings`*, (c)po spustení servicu* `Start Tracking`*, (d)pri posielaní dát cez internet*

Dôležitou súčasťou tejto aplikácie je, aby sme zariadenie počas trackovania (zbierania dát) udržali stále aktívne (v prebudenom stave). V tom nám pomôže trieda PowerManager [20]. Je potrebné si dávať pozor, pretože využitie tohoto API výrazne ovplyvní spotrebu batérie zariadenia. Bez použitia tejto triedy sa nám pri testovaní stávalo, že procesor prešiel do režimu spánku a naša aplikácia prestala zbierať dáta. Preto je táto časť nevyhnutnou súčasťou aplikácie, ktorá musí bežať nepretržite na pozadí. Inštanciu tejto triedy získame príkazom `Context.getSystemService(Context.POWER_SERVICE)` a hlavné API, ktoré využívame je `newWakeLock()`. Tento príkraz vytvorí objekt `PowerManager.WakeLock` [21], ktorého metódami ovládame stav zariadenia. Hlavné metódy, ktoré sa využívajú, sú `acquire()` a `release()`. Metóda `acquire()` získa WakeLock a donúti zariadenie ostať v bdelom stave (podľa úrovne, ktorú máme nstavenú). V momente, keď už nepotrebujeme držať zariadenie v bdelom stave, využijeme metódu `release()`, a teda uvoľníme WakeLock (je to potrebné urobiť čo najskôr, aby sme sa vyhli zbytočnému strácaniu energie z batérie zariadenia). Pri vytváraní WakeLock si vždy definujeme úroveň prebudenia:

Hodnota parametra	*CPU*	*Obrazovka*	*Klávesnica*
`PARTIAL_WAKE_LOCK`	ON*	OFF	OFF
`SCREEN_DIM_WAKE_LOCK`	ON	stlmená jasnosť	OFF
`SCREEN_BRIGHT_WAKE_LOCK`	ON	jasná	OFF
`FULL_WAKE_LOCK`	ON	jasná	jasná

*Pri držaní WakeLock, CPU bude stále bežať (aj keď sa vypne obrazovka alebo užívateľ prepne zariadenie do režimu spánku tlačidlom ZAPNÚŤ/VYPNÚŤ). V ostatných typoch WakeLock sa zariadenie po stlačení tlačidla ZAPNÚŤ/VYPNÚŤ prepne do režimu spánku.

Pre vytváranie a používanie PowerManagera je potrebné získať povolenia v `AndroidManifest.xml`. V našej GPS aplikácii používame `PARTIAL_WAKE_LOCK`. WakeLock získame po inicializovaní requestov na LocationManager pri metóde `onStartCommand()` alebo service beží a zistíme zapnutie GPS. Uvoľníme ho v momente, keď zistíme vypnutie GPS alebo pri zastavení celého servicu v metóde `onDestroy()`. Teda keď zbierame GPS dáta, udržujeme CPU zariadenia v bdelom stave. Keď je GPS vypnuté a nezbierame dáta, ostáva naša aplikácia v nečinnom stave, a teda nemusíme nútiť CPU do bdelého stavu.

```
//AndroidManifest.xml
<uses-permission android:name="android.permission.WAKE_LOCK" />

//GPStracker.java service
PowerManager pm = (PowerManager) this.getSystemService(Context.POWER_SERVICE);
PowerManager.WakeLock wl= pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, TAG);
wl.acquire();
//...CPU ostva  aktvne  v tejto asti
wl.release();
```

Po vysvetlení hlavnej funkcionality aplikácie na zbieranie GPS polohy zhrnieme, ako presne aplikácia funguje. Vždy, keď stlačíme tlačidlo Start_Tracking, vytvorí sa service (ak ešte neexistoval) GPSTracker a do súboru si dáme značku, že začína nový „track“ slovom „RESET“ (túto značku si vytvoríme vždy, keď je vypnuté GPS alebo zastavený celý service tlačidlom Stop_Tracking). Inicializujeme BroadcastReceiver na určovanie pripojenia na internet, LocationListener s jeho príslušným LocationManager a ak je GPS zapnuté, získame WakeLock. Pri zmene lokácie si zapamätáme lokáciu v globálnom ArrayListe - v tvare čas, latitude, longitude. Ak tento zoznam presiahne veľkosť 10 bodov, všetky body zapíšeme do lokálneho súboru (názov súboru je ID androida zariadenia, kvôli jedinečnosti názvu súboru v rámci všetkých zariadení pri posielaní na server[5]) a ArrayList vynulujeme. Pri používaní ArrayListu a pri práci so súborom používame uzamykacie premenné a pomocou metódy `synchronized(lock)` ošetríme výnimky, kde by naraz chcela aplikácia čítať aj zapisovať do ArraListu alebo súboru. Ak súbor presiahne veľkosť 3kB alebo bolo vypnuté GPS (teda berieme to ako koniec „tracku“) a sme pripojení na internet, tak vytvoríme AsyncTask a pripojíme sa na server. Pošleme dáta a ak posielanie skončilo bez výnimky (bolo úspešné), tak dáta z lokálneho súboru vymažeme. Ak nie sme k internetu pripojení, tak sa súbor pokúsime poslať, keď dostaneme notifikáciu o pripojení na internet.

2.2 Predspracovanie nazbieraných surových GPS dát

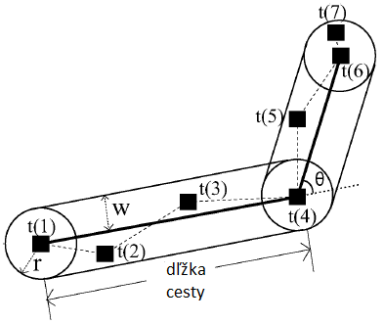
V [12][1] navrhli predspracovanie dát pred analýzou vlastností získaných „trackov“. Keďže sa človek nepohybuje úplne po priamke medzi dvoma miestami pred výraznou zmenou smeru ($\pm 2 metre$), tak bude správnejšie vyhodnotiť tento pohyb priamočiaro dlhou cestou, ako nie-

[5] `android.provider.Settings.Secure.getString(getContentResolver(), android.provider.Settings.Secure.ANDROID_ID)`

koľko malých ciest s drobnou zmenou smeru. Preto využijeme dve metódy predspracovania dát[12]:

- oblžníková „rectangular“ metóda - vezmeme si tri po sebe idúce body. Okolo bodu 1 a 3 si vytvoríme kružnicu s polomerom r. Vytvoríme si dotyčnicu zhora a zdola. Ak nám bod 2 padne do vnútra tejto oblasti, tak tento bod vymažeme a cestu spravíme z bodu 1 do bodu 3. Pokračujeme s bodom 1, bodom 3 a bodom 4. Programovo sme to implementovali najkratšou vzdialenosťou bodu 2 od úsečky |13|. Ak bola táto vzdialenosť väčšia ako r, tak bod 2 ponecháme a v ďalšom kroku berieme body 2,3,4. V opačnom prípade bod 2 vymažeme. Na príklade Obr. 2.3 vo výslednom tracku ostanú body 1,4,7, ostatné body sú vymazané.
- uhlová „angle“ metóda - obdlžníková metóda nezachytí prípady, keď ideme približne rovnakým smerom, ale cesty sú dlhé (vzdialenosť bodu od úsečky môže byť veľká). Preto si vezmeme tri body a počítame uhol pri strednom bode. Ak je tento uhol θ väčší ako zvolený parameter a_θ, potom tento bod vyhodíme a cesta bude úsečka medzi bodom 1 a 3. Pokračujeme s ďalšími bodmi ako v minulej metóde(Obr. 2.3).

V našej simulácii sme na track použili najprv obdlžníkovú metódu s parametrom $r = 2m$ a dostali sme track_1. Následne na track_1 sme použili uhlovú metódu s parametrom $\theta = 160°$. Pozerali sme sa tiež na rýchlosť pohybu počas jednotlivých ciest a ak táto rýchlosť prevyšovala hodnotu 55 m/s (180 km/h), tak sme tento „track“ rozdelili v tomto bode (s najväčšou pravdepodobnosťou je to diskrepancia v GPS lokácii). Brali sme do úvahy len „tracky“ s aspoň 10 bodmi. Takto sme dostali výsledné predspracované dáta vhodné na analýzu vlastností pohybu.



Obr. 2.3: *Znázornenie obdlžníkovej a uhlovej metódy [12]*

Kapitola 3

Analýza získaných dát a návrh modelu pohybu

V tejto kapitole sa budeme venovať popisu a analýze vlastností, ktorými je konkrétny pohyb charakterizovaný. Teda konkrétny „track“, ako zoznam GPS bodov, sme zjednodušili do niekoľkých vlastnotí, ktoré popisujú pohyb v „tracku“. Následne sme navrhli model, ktorý pre pohyb využíva vlastnosti, čo najpodobnejšie získaným reálnym dátam.

3.1 Vlastnosti pohybu sledované na reálnych dátach

Modely sú podľa definície iba aproximácie neznámej reality alebo skutočnosti. George Box vyslovil známe tvrdenie: „Všetky modely sú zlé, ale niektoré sú užitočné“. Pre vytvorenie užitočného modelu sa budeme v tejto kapitole venovať analýze zozbieraných dát pomocou GPS aplikácie (podkapitola 2.1), ktoré boli následne upravené do jednoduchšej formy (podľa podkapitoly 2.2). GPS aplikáciu nosilo 42 dobrovoľníkov a nazbierali spolu 948 „trackov“. Dáta máme vo forme zoznamu bodov v tvare čas, latitude, longitude. Všetky body predstavujú jeden presun tzv. „track“. Budeme analyzovať vlatnosti „jednotlivá cesta“, „podmienený uhol“ a „rozptyl“ pre všetky zozbierané „tracky“. Následne informácie o týchto vlastnostiach využijeme pri navrhovaní nového algoritmu pre pohyb (syntetický model).

Analýzu dát tiež využijeme pri voľbe správnych parametrov pre simuláciu umelých algoritmov z kapitoly 1.

3.1.1 Jednotlivá cesta (vzdialenosť)

Jedná sa o vzdialenosť medzi dvoma po sebe idúcimi bodmi lokácie. Zo všetkých „trackov“ vezmeme vzialenosti medzi každými dvoma bodmi, čím dostaneme empirickú distribúciu vzdialeností, ktoré dobrovoľníci prešli pred zmenou smeru. Pokúsili sme sa vybrať známu parametrickú spojitú distribúciu, ktorá sa najviac podobala na získanú empirickú distribúciu. Z prác [10][11][12], ktoré sledovali túto vlastnosť sme očakávali, že sa bude jednať o „heavy-tailed“ distribúciu, ale brali sme do úvahy aj iné parametrické distribúcie, napr. Gaussovu, logistickú, gamma distribúciu,... Pri výbere najvhodnejšej spojitej funkcie sme využili Akaike informačné kritérium v kombinácii s MLE[1].

MLE odhad

MLE je spôsob určenia najlepších hodnôt pre parametre distribúcie. V praxi to funguje nasledovne: zobereme distribúciu s neznámym parametrom θ. Pozorujeme náhodnú premennú X. Vierohodnosť (likelihood) parametra θ pri pozorovaní náhodnej premennej X je

$$L(X|\theta) = Pr(\theta|X)$$

Hodnota θ, ktorá maximalizuje vierohodnosť je najvierohodnejší odhad $\widehat{\theta}$. Väčšinou na zistenie hodnoty $\widehat{\theta}$ zderivujeme pravdepodobnostnú funkciu, čím získame hodnotu s maximom funkcie [25].

Akaike informačné kritérium

Akaike informačné kritérium (AIC) [26] je nástroj na kvantitatívne ohodnotenie blízkosti (podoby) vybranej spojitej distribúcie a empirických dát x. AIC je založené na Kullback-Leibler (K-L) informácii. Vezmime si koncept, kde f označuje realitu a g predstavuje model, ktorý túto realitu aproximuje (pravdepodobnostná distribúcia). K-L informácia $I(g, f)$

[1] „maximum likelihood estimation“ - odhadovanie maximálnej vierohodnosti

určuje množstvo stratenej informácie, ak by sme aproximačným modelom g simulovali realitu f. Pri spojitých funkciach by sme dostali integrál:

$$I(f,g) = \int f(x) log\left(\frac{f(x)}{g(x|\theta)}\right) dx$$

Najlepší model stratí najmenej informácie. Ak máme prístup k celej realite f, jej hodnota sa nemení. Našou úlohou je teda minimalizovať hodnotu $I(f,g)$ prechádzaním cez priestor g zmenou parametra θ. Často však nemáme prístup k celej realite f a k najlepšiemu parametru θ, preto K-L informácia pracuje so strednými (očakávanými) hodnotami:

$$I(f,g) = E_f[log(f(x))] - E_f[log(g(x|\theta))]$$

Hodnota $E_f[log(f(x))]$ je pri prechádzaní rôznymi modelmi rovnaká (konštanta C), preto nás zaujíma hlavne hodnota $E_f[log(g(x|\theta))]$. Akaike (1973, 1974, 1985, 1994) našiel formálny vzťah medzi K-L informáciou a teóriou vierohodnosti („likelihood theory“). Hodnotu θ nahradil najvierohodnejším odhadom MLE $\widehat{\theta}$ pre konkrétnu vzorku dát reality y:

$$E_y E x[log(g(x|\widehat{\theta}(y)))]$$

Ukázal , že túto strednú hodnotu môžeme približne ohodnotiť zlogaritmizovanou funkciou vierohodnosti vychýlenej o parameter K (počet parametrov pre model g). Akaike rátal funkciu (prenásobenú hodnotou -2 z historických dôvodov):

$$AIC = -2log(L(\widehat{\theta}|data)) + 2K$$

Keďže sa berie do úvahy počet parametrov modelu (distribúcie), zabraňuje sa tak „overfittingu[2]“, ktorý môže nastať pri nekompletnej vzorke dát o realite.

Na analýzu sme využili nástroj Matlab 7.10.0(R2010a). Matlab má v sebe implementovanú funkciu `fitdist(data, distname)` [27], ktorá využíva MLE na nájdenie najvhodnejších parametrov distribúcie `distname`. Následne sme využili funkciu `allfitdist(data,`

[2]pre vzorku reality x, funkcia skáče medzi hodnotami dát x, ktoré nám boli poskytnuté a medzi nulou - x osou pre dáta, ktoré sa nedostali do vzorky reality x. Dochádza k tzv. preučeniu.

`sortby, varargin)`, ktorú spísal Mike Sheppard v roku 2012 a stala sa na MatlabCentral blogu najlepším príspevkom týždňa [28]. Táto funkcia vezme dáta a konkrétnu distribúciu a pomocou funkcie `fitdist()` najprv priradí distribúcii hodnoty parametrov, potom zráta hodnotu AIC kritéria. Vyskúša rôzne distribúcie a usporiada výstup podľa hodnoty AIC. Môžeme si vybrať, či chceme testovať distribúcie diskrétne alebo spojité, či chceme zobraziť graf PDF(pravdepodobnostnej distribučnej funkcie) alebo CDF(kumulatívnej distribučnej funkcie) a podľa akého kritéria chceme distribúcie zoradiť od najlepšej po najhoršiu (zvoleného podľa parametra `sortby`, napr. AIC, BIC[3] (default), negative loglikelihood, AICc[4]). Distribúcie, ktoré funkcia testuje sú: Beta, Birnbaum-Saunders, Exponential, Extreme value, Gamma, Generalized extreme value, Generalized Pareto, Inverse Gaussian, Logistic, Log-logistic, Lognormal, Nakagami, Normal, Rayleigh, Rician, t location-scale, Weibull. Ako výstup dostaneme pre každú distribúciu štruktúru s hodnotami najlepšie sa hodiacich parametrov a hodnotu AIC spolu s grafom empirických dát a najlepších spojitých parametrických distribúcii.

Celkovo sme nazbierali 48941 jednotlivých ciest. Tieto cesty naozaj vykazovali „heavy-tailed“ distribúciu, pričom ako najlepšiu sme si vybrali Generalized Pareto[5] distribúciu. Pre $k > 0$ a $\theta < x$ dostaneme pravdepodobnostnú funkciu:

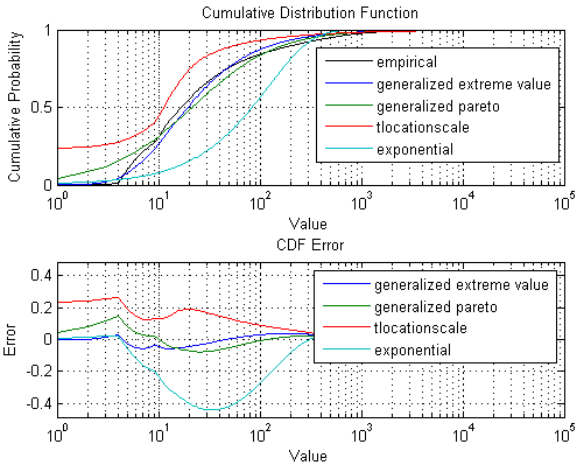
$$f(x|k,\sigma,\theta) = \left(\frac{1}{\sigma}\right)\left(1 + k\frac{(x-\theta)}{\sigma}\right)^{-1-\frac{1}{k}}$$

Hodnoty parametrov boli: tvar $k = 0.9020$, umiestnenie (škálovanie) $\sigma = 22.2567$ a threshold $\theta = 0$. V skutočnosti najlepšie dopadla generalized extreme value distribúcia, ale keďže sme upravovali dáta obdlžníkovou a uhlovou metódou, málokedy sa tam vyskytovali malé hodnoty vzdialeností. Lepšie preto dopadla distribúcia s „heavy-tailed“ chvostom na oboch stranách. Pre naše potreby nám ale postačí Pareto distribúcia Obr. 3.1a. Túto distribúciu sme pri simulácii využili pri Levy-Walk modeli a našom ACRM modeli (podkapitola 3.4). Pri simulácii sme Pareto distribúciu ohraničili z dolnej strany 23 metrami ako minimum (dôležité kvôli parametru σ) a zhora maximálnou hodnotou vdialenosti 3000 (aby sme nerobili dlhé cesty pri premiestnení sa medzi dvoma miestami v jednom „tracku“).

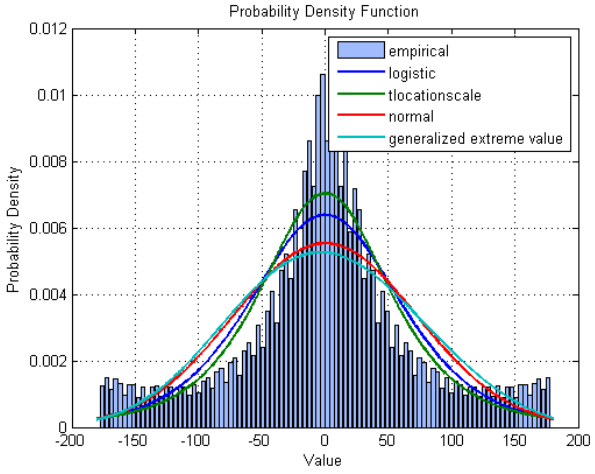
[3] Bayesovo informačné kritérium

[4] AIC upravené pre malú vzorku dát

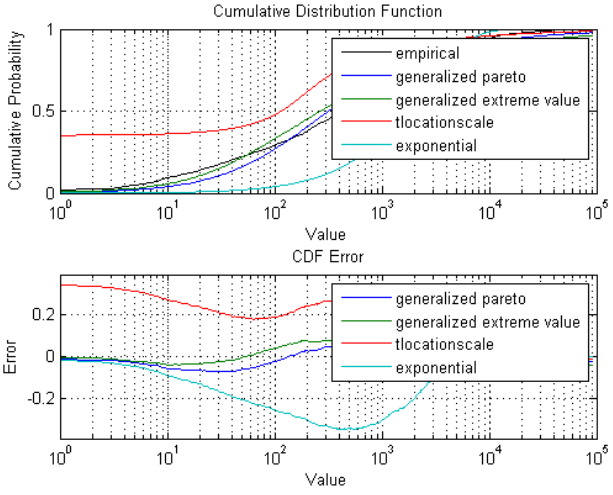
[5] http://www.mathworks.com/help/stats/generalized-pareto-distribution.html



(a) Jednotlivé cesty



(b) Podmienený uhol



(c) Rozptyl

Obr. 3.1: *Vlastnosti reálneho pohybu*

3.1.2 Podmienený uhol

Podmienený uhol predstavuje uhol odklonenia od správneho smeru. Za správny smer považujeme vždy smerovanie ku poslednému bodu tracku. Vezmime si ľubovoľný bod v tracku (bod 1), jeho nasledujúci bod (bod 2) a koniec tracku (bod 3). Podmienený uhol je uhol zvieraný úsečkami |12| a |13|. Pre analýzu budeme brať tento uhol v kladnej aj zápornej forme, teda nezáleží na smere odklonenia. Tento uhol sme rátali pre surové dáta, aby pri distribúcii nedošlo ku skresleniu (napr. ak ideme dlho jedným smerom, uhol 10° odklonenia sa v surových dátach vyskytuje 10 krát, ale po úprave dát iba raz). Celkovo počítame s 486210 odkloneniami. Podľa AIC kritéria najvhodnejšia distribúcia bola logistic[6] distribúcia (distribúcia podobná Gaussovej distribúcii, ale s ťažšími chvostami) s parametrami priemer $\mu = 0$ a škálovanie $\sigma = 39.1463$ (Obr.3.1b). Pravdepodobnostná funkcia pre $\sigma \geq 0$ a $-\infty < x < \infty$:

$$f(x|\mu,\sigma) = \frac{\exp\{\frac{x-\mu}{\sigma}\}}{\sigma\left(1+\exp\{\frac{x-\mu}{\sigma}\}\right)^2}$$

Minimálny uhol si pri simulácii modelu ACRM ohraničíme na $-180°$ a maximum na $180°$.

3.1.3 Rozptyl

Rozptylom nazývame vzdialenosť medzi začiatočným a koncovým bodom tracku pre horizontálny smer - latitude a pre vertikálny smer - longitude. Pri analýze sme využívali iba ciele do 10 km, keďže takýchto rozptylov bolo takmer 95%. Znova sa ako najlepšia ukázala Generalized Pareto distribúcia (Obr. 3.1c) s parametrami tvar $k = 1.3486$, umiestnenie (škálovanie) $\sigma = 244.9597$ a threshold $\theta = 0$.

3.1.4 Čas

Vlastnosť čas využijeme pri simulácii umelých algoritmov pre určenie doby simulácie. Pri porovnávaní reálnych a simulovaných „trackov“ chceme, aby boli porovnávané vlastnosti nezávislé na dĺžke simulácie umelých algoritmov. Preto si zoberieme celkový čas pre každý „track“ a takúto dobu budeme simulovať syntetické modely pohybu opísané v kapitole 1. Ak by bola dĺžka zvolenej poslednej jednotlivej cesty dlhšia ako nám dovoľuje prejsť čas

[6] http://www.mathworks.com/help/stats/logistic-distribution.html

a rýchlosť, tak túto cestu usekneme podľa toho, koľko máme času. Priemerný čas jedného „tracku“ pre reálne dáta bol 52,155 minúty.

3.2 ACRM „Aimed Constrained Random Movement“ model

Existuje viacero modelov, ktoré berú do úvahy mnoho faktorov ovplyvňujúcich náš pohyb, napr. spoločenský život, obľúbené destinácie, konkrétna mapa pohybu,... Avšak s každým ďalším parametrom sa matematický popis (model) pohybu a predpoklad pravdepodobnosti ďalšieho možného pohybu komplikuje. Aby tieto algoritmy fungovali správne potrebujeme nastaviť správne parametre. V mnohých situáciach však nemáme a nepoznáme najlepšie hodnoty pre správnu simuláciu takýchto algoritmov. My sme sa preto snažili navrhnúť algoritmus, ktorý by bol jednoduchý (mal čo najmenej parametrov a bol dobre matematicky popísateľný) a zároveň, aby sa čo najviac podobal na reálny pohyb. Navrhli sme správanie modelu podľa skúseností a pozorovaní reálneho pohybu. Parametre algoritmu sme nastavili podľa analýzy vlastností reálnych „trackov“.
Model funguje nasledovne:
Vyberieme si cieľ, kam máme namierené. Z toho vyplýva vlastnosť „Aimed“, teda cielený pohyb. Vyberáme latitude a longitude z distribúcie získanej z vlastnosti „rozptyl“ (podkapitola 3.1.3), teda ohraničená Generalized Pareto distribúcia. Z toho vyplýva vlastnosť „Constrained“, teda ohraničený pohyb. Naša cesta k cieľu však nebude priamočiara, ale bude sa skladať z jednotlivých ciest. Vždy si vyberieme dĺžku cesty z Pareto distribúcie získanej z vlastnosti „jednotlivé cesty“ (podkapitola 3.1.1). Povolíme mierne odklonenie od smeru k cieľu posunutím o uhol z logistic distribúcie vlastnosti „podmienený uhol“ (podkapitola 3.1.2). Z týchto krokov algoritmu vyplýva vlastnosť „Random“, teda náhodný pohyb. Celkovo dostaneme cielený ohraničený náhodný model pohybu - ACRM model.
Rýchlosť pohybu nastavíme podobne ako v modeli Levy Walk (podkapitola 1.7), teda čas určený na presun z jedného bodu do nasledujúceho bodu závisí od dĺžky cesty a vhodne nastavených parametrov.
Ak sme v mieste (x_{n-1}, y_{n-1}), nasledujúci bod umiestnenia (x_n, y_n) v tracku získame nasledovne:

$$x_n = x_{n-1} + \cos(360 + a + a_n)d_n$$

$$y_n = y_{n-1} + \sin(360 + a + a_n)d_n$$

a - uhol ku vybranému cieľu na jednotkovej kružnici so stredom v bode (x_{n-1}, y_{n-1})
a_n - odklonenie od smeru ku cieľu, náhodná premenná z distribúcie podmieneného uhlu
d_n - dĺžka cesty, náhodná premenná z distribúcie jednotlivých ciest

Celý „track“ skončí po uplynutí určitého času (príp. môžeme nastaviť koniec „tracku“ po príchode do dostatočne blízkej vzdialenosti ku cieľu).
Tento model je pružný a vhodný aj pre náročnejších používateľov. Podľa potreby tam dokážeme implementovať zložitejšie vlastnosti pohybu, napr. obmedzíme výber cieľa podľa predom stanovených pravidiel (obľúbené lokality, pohyb iných mobilných uzlov,...), môžeme ovplyvniť, ako veľmi priamočiaro sa chceme ku cieľu pohybovať pomocou nastavenia parametrov distribúcie pre výber dĺžok jednotlivých ciest, distribúcie pre podmienený uhol odklonenia, zmena rýchlosti pohybu (ak by sme napríklad chceli pracovať s modelom pohybu bez dopravných prostriedkov) a taktiež môžeme nastaviť celodenný pohyb pomocou výberu niekoľkých cieľov. Mohli by sme teda povedať, že je to tzv. prechodný model medzi syntetickými a sledovanými modelmi (podkapitola 1.2). V jeho najjednoduchšej forme vykazuje lepšie výsledky ako Random Walk model alebo Random Waypoint a veľmi jednoducho vieme tento model obmeniť nastavením viacerých parametrov na sledované modely pre konkrétne situácie. Pri analýze porovnávania úspešnosti jednotlivých syntetických modelov budeme pracovať s jednoduchou formou ACRM modelu a parametre nastavíme podľa analýzy z reálnych nazbieraných dát (Obr. 3.2 - červená čiara predstavuje smer od začiatku pohybu k cieľu).

Obr. 3.2: *ACRM „Aimed Constrained Random Movement“ model*

Kapitola 4

Porovnanie syntetických modelov a reálnych dát

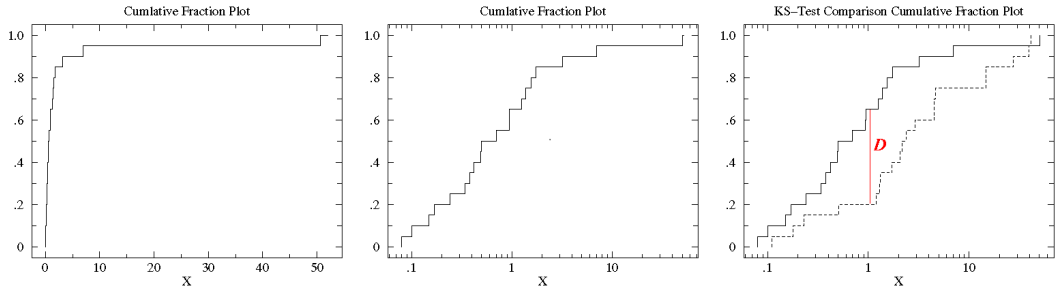
Pre každý reálny „track“ sme simulovali 5 syntetických modelov z kapitoly 1 a náš ACRM model z kapitoly 3. Pre každý syntetický model sme nasimulovali 948 „trackov“ a všetky modely mali rovnaký čas simulácie ako reálne „tracky“. Začiatok simulácie sme vždy vybrali náhodne z plochy 5000m x 5000m a následne sa pohybovali podľa navrhnutého modelu. Pre simuláciu sme modely naprogramovali v jazyku Java. Pomocou knižnice JSC (Java Statistical Classes) [32] sme mohli pracovať s rôznymi pravdepodobnostnými distribúciami (Pareto, normal, logistic, ...). Všetky zdrojové kódy (pre simuláciu modelov aj pre získavanie vlastností z „trackov“) sme priložili na DVD v prílohe A.

Na porovnanie reálnych dát a simulovaných modelov využijeme dvojvýberový Kolmogorov-Smirnov test.

4.1 Dvojvýberový Kolmogorov-Smirnov test

Vezmime si dáta nazbierané v dvoch odlišných situáciach (pre náš prípad reálne dáta a simulované dáta). Vzorka pre reálne dáta $X_1, X_2, ..., X_n$ má distribúciu s cdf[1] F(x) a vzorka pre simulované dáta $Y_1, Y_2, ..., Y_n$ má distribúciu s cdf G(x). Chceme porovnať ako veľmi sa

[1] kumulatívna distribučná funkcia



(a) CDF pre dáta X

(b) CDF pre dáta X logaritmicky naškálované

(c) CDF pre dáta X a Y s K-S2 hodnotou

Obr. 4.1: *Kumulatívna distribučná funkcia (CDF)*

tieto distribúcie podobajú. Dvojvýberový Kolmogorov-Smirnov test (K-S2) je neparametrický test na porovnanie dvoch množín dát z kumulatívnych distribučných funkcií (dvoch empirických distribúcií). Základná hypotéza je, že dáta sú v oboch množinách z rovnakej distribučnej funkcie (o distribúcii nerobí žiadne predpoklady).

$$H_0 : F = G vs. H_1 : F \neq G$$

Je to jeden z najpoužívanejších testov na porovnanie dvoch vzoriek dát, keďže je citlivý na zmenu umiestnenia aj tvaru empirickej kumulatívnej distribučnej funkcie [31].
Príklad [29] : Pre dáta $X = \{0.08, 0.10, 0.15, 0.17, 0.24, 0.34, 0.38, 0.42, 0.49, 0.50, 0.70, 0.94, 0.95, 1.26, 1.37, 1.55, 1.75, 3.20, 6.98, 50.57\}$ si graficky znázorníme ako sú distribuované cdf, Obr.4.1a. Evidentne žiadne dáta neležia pod hodnotou 0.08, 5% dát je nižších ako 0.10, 10% je nižších ako 0.15, 15% dát je nižších ako 0.17. 17 bodov z 20 je nižších ako π, preto môžeme povedať, že „cumulative fraction“, dát menších ako π, je 85%. Pre akékoľvek číslo x, „cumulative fraction“ je percentuálna časť dát, ktorých hodnota je menšia[2] ako hodnota x (Obr.4.1a).

V našom prípade je väčšina dát nahromaždená v ľavej časti grafu. To je znak, že sa nejedná o normálnu (Gaussovu distribúciu). Pre lepšie znázornenie rozdielov medzi malými číslami x môžeme použiť logaritmické naškalovanie, Obr.4.1b. Logaritmické naškalovanie môžeme použiť vďaka tomu, že všetky naše hodnoty sú kladné čísla. Tu vidíme, že medián je niečo pod 1. Na Obr. 4.1c sú znázornené „cumulative fraction“ funkcie pre dáta X a dáta $Y =$

[2]niekde rátaju počet hodnôt striktne menších, niekde menších rovných

{2.37, 2.16, 14.82, 1.73, 41.04, 0.23, 1.32, 2.91, 39.41, 0.11, 27.44, 4.51, 0.51, 4.50, 0.18, 14.68, 4.66, 1.30, 2.06, 1.19}. K-S2 test ráta hodnotu D, hodnotu maximálneho výškového rozdielu medzi týmito funkciami. V našom prípade je $D = 0.45$ (rozdiel medzi „cumulative fraction“ hodnotami oboch vzoriek v tej istej hodnote x).

$$D = max_x(|F(X) - G(X)|)$$

Hodnota D, tiež nazývaná štatistický výsledok, nie je ovplyvnená logaritmickým škálovaním (len presunie miesto dôležitého regiónu, hodnotu x, kde sa D nachádza). Následne si vypočítame asymptotickú hodnotu p z intervalu $(0, 1)$. p je pravdepodobnosť, že vzdialenosť distribúcie X a Y je väčšia alebo rovná ako sme odsledovali K-S2 testom z poskytnutých dát (vzoriek). Čím menšia je hodnota p, tým vzdialenejšie sú od seba porovnávané distribúcie. Ak je hodnota p malá, môžeme predpokladať, že distribúcie pre X a Y boli rôzne. Kolmogorov-Smirnov test nám vráti hodnotu h, ktorá zamietne alebo prijme hypotézu pri 5% úrovni dôležitosti (táto hodnota sa porovnáva s hodnotou p).
V prostredí Matlab sme využili integrovanú funkciu pre dvojvýberový Kolmogorov-Smirnov test `[h,p,D] = kstest2(X,Y)` [30].

4.2 Porovnávané vlastnosti

Budeme porovnávať vlastnosti „trajektória“, „počet susedov“ a „celkové pokrytie“. Sú to dôležité vlastnosti ovplyvňujúce úspešnosť routovacích protokolov pre simulovaný pohyb mobilných uzlov.

4.2.1 Trajektória

Trajektória je celková prejdená vzdialenosť v jednom „tracku“ (nasčítané všetky jednotlivé cesty v „tracku“). Ak si vyhodnotíme trajektóriu pre všetky „tracky“ modelu dostaneme empirickú distribúciu trajektórii modelu. Distribúciu syntetického a reálneho modelu sme porovnali K-S2 testom:

model	*p*	*D*	*priemer*
`reálne dáta`			6.2247e+003
`ACRM model`	1.2533e-009	0.1486	7.9977e+003
`Levy Walk`	3.0058e-008	0.1370	7.9143e+003
`Prb. Chiang`	4.2732e-062	0.3846	2.1882e+004
`Gauss-Markov`	1.1121e-091	0.4679	3.1586e+004
`Random Walk`	2.1395e-069	0.4067	2.6336e+004
`Random Waypoint`	5.9721e-098	0.4837	3.4074e+004

Keďže „tracky“ majú pri všetkých modeloch rovnaký čas simulácie, dĺžka trajektórie bude závisieť iba od rýchlosti využitej pri pohybe. V Levy Walk a ACRM modeli sme použili rovnaký algoritmus pre výpočet použitej rýchlosti (taktiež sme použili rovnakú distribúciu pre výber dĺžky jednotlivej cesty, od ktorej výpočet rýchlosti závisí), preto tieto dva modely vykazovali podobné výsledky. Ostatné modely mali v priemere 10-krát dlhšie trajektórie a aj v K-S2 teste vykazovali oveľa horšiu podobu na reálne dáta. Teda zohľadnenie dĺžky jednotlivej cesty pre výpočet rýchlosti, ako sme počítali rýchlosť v Levy Walk a ACRM modeli, je omnoho lepšie ako pevné stanovenie jednotnej rýchlosti pre celú simuláciu (pravdepodobnostný Chiangov model) alebo výber rýchlosti z uniformnej distribúcie (Random Walk, Random Waypoint).

4.2.2 Počet susedov

Dva mobilné uzly považujeme za susedov, ak vzdialenosť medzi nimi nie je väčšia ako parameter variance - dosah signálu pre vzájomnú komunikáciu medzi mobilnými uzlami. Počet susedov, s ktorými dokáže uzol komunikovať, je jedna z najdôležitejších vlastností pre návrh routovacích protokolov.
Implementácia: všetky „tracky“ namapujeme do jednej oblasti v ľavom dolnom rohu (nájdeme minimálnu latitude a longitude a všetky body tohto „tracku“ posunieme o tieto parametre v zápornom smere, aby sme sa dostali do oblasti $(0, 0)$ až $(max\_latitude, max\_longitude)$). *Tento krok nie je potrebný, ak máme dostatok dát získaných (simulovaných) v ohraničenej oblasti, napr. školský areál.* Následne simulujeme pohyb mobilného uzla istý čas (5min, 15min, 30 min, 60min) a po uplynutí času sa vždy pozeráme na počet mobilných uzlov v mojom okolí. Dostali sme príslušné empirické distribúcie, ktoré sme porovnali s distribúciou pre reálne dáta K-S2 testom:

model	*p_05min*	*D_05min*	*p_15min*	*D_15min*
`ACRM model`	5.8092e-005	0.1043	0.0379	0.0643
`Levy Walk`	3.1216e-006	0.1180	6.8370e-008	0.1338
`Prb. Chiang`	4.5423e-020	0.2171	1.7277e-024	0.2403
`Gauss-Markov`	2.6011e-027	0.2540	1.9408e-028	0.2592
`Random Walk`	1.0886e-019	0.2150	3.1355e-021	0.2234
`Random Waypoint`	1.3730e-029	0.2645	1.9164e-032	0.2771
model	*p_30min*	*D_30min*	*p_60min*	*D_60min*
`ACRM model`	0.0332	0.0653	0.0026	0.0832
`Levy Walk`	3.0058e-008	0.1370	1.6909e-009	0.1475
`Prb. Chiang`	7.5719e-018	0.2044	2.1688e-018	0.2076
`Gauss-Markov`	1.0928e-032	0.2782	3.3575e-036	0.2929
`Random Walk`	2.9248e-020	0.2181	2.0046e-022	0.2297
`Random Waypoint`	9.2942e-062	0.3836	2.8695e-067	0.4004

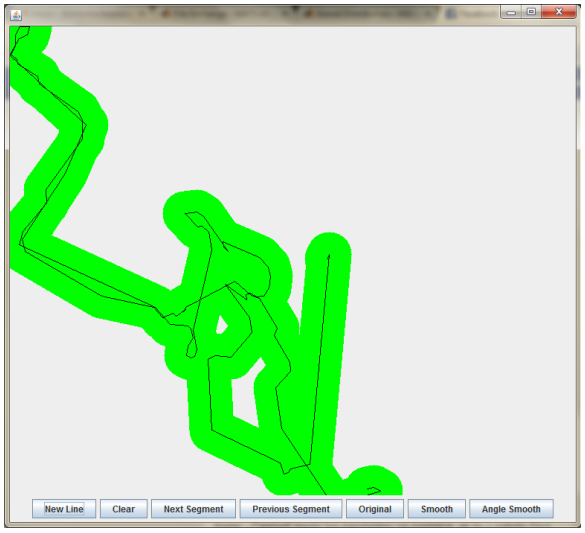
Najlepšie dopadol podľa K-S2 testu ACRM model s výrazne lepšími hodnotami p ako ostatné syntetické modely. Len pre porovnanie priemerná hodnota susedov pre 30 minút je pre reálne dáta neigh=27.2329 , pre ACRM model neigh=25.0685, pre Levy Walk neigh=14.4995 a pre Random Waypoint neigh=4.1012. Teda pri rôznych modeloch dochádza k veľkému skresleniu počtu susedov v okolí uzla a v dôsledku toho, bude úspešnosť použitých routovacích protokolov výrazne závisieť od výberu modelu pre simuláciu pohybu mobilných uzlov.

4.2.3 Celkové pokrytie

Mobilný uzol v konkrétnom mieste (x, y) pokrýva oblasť kruhu so stredom (x, y) a polomerom parametra *variance*. Teda pokrýva oblasť, kde každý ďalší mobilný uzol je mojim susedom a môžeme komunikovať. Prienikom oblastí počas celej prejdenej trajektórie „tracku“ získame celkové pokrytie (Obr.4.3). Táto vlastnosť je dôležitá z pohľadu opakovaného stretávania konkrétnych mobilných uzlov. Ak mobilný uzol robí malé pokrytia, tak je veľká šanca, že sa vracia opakovane na rovnaké miesta. Ak robí veľké pokrytia, pravdepodobnosť, že sa vráti na miesto (x, y) v krátkom časovom intervale, je veľmi malá. V praxi to znamená: ak sa mobilný uzol A stretol s mobilným uzlom B v oblasti (x, y), vieme podľa našich odhadovaných pokrytí povedať, aká je šanca stretnúť sa znova (pri malých

pokrytiach je šanca stretávať sa opakovane výrazne vyššia).

Vlastnosť celkového pokrytia sme implementovali bit-mapou veľkosti ($max\_latitude, max\_longitude$). Následne sme prechádzali jednotlivé cesty „tracku“ a pozreli sa na pokrytie cesty, teda na obdlžníkovú oblasť s ľavým horným rohom $min(X_{n-1} - variance, X_n - variance)$, $min(Y_{n-1} - variance, Y_n - variance)$ a pravým dolným rohom $max(X_{n-1} - variance, X_n - variance)$, $max(Y_{n-1} - variance, Y_n - variance)$. Pre každý bod v oblasti spočítame vzdialenosť od cesty $min(X_{n-1}, Y_{n-1}, X_n, Y_n)$. Ak je táto vzdialenosť menšia ako $variance$, v bit-mape nastavíme pre tento bod hodnotu $true$ - tento bod je v pokrytí cesty. Na konci prejdeme celú bit-mapu a spočítame koľko bodov sa nachádzalo v celkovom pokrytí „tracku“.

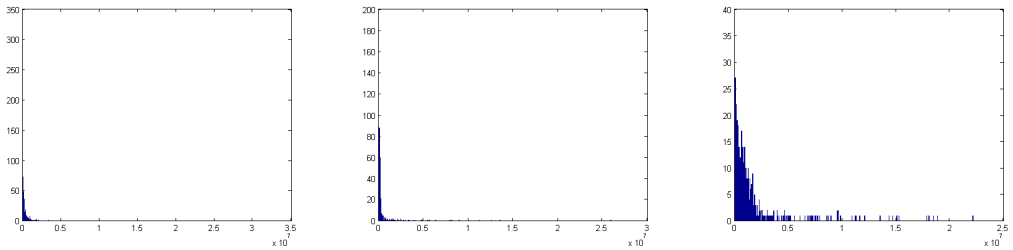


Obr. 4.2: *Celkové pokrytie pre reálny „track“*

Porovnaním distribúcií K-S2 testom sme dostali:

model	*p*	*D*
`ACRM model`	2.1029e-011	0.1623
`Levy Walk`	2.2766e-009	0.1465
`Prb. Chiang`	1.0615e-024	0.2413
`Gauss-Markov`	1.6888e-037	0.2982
`Random Walk`	6.3811e-034	0.2835
`Random Waypoint`	8.2059e-102	0.4932

V tomto teste sa najlepšie umiestnil Levy Walk model. Na druhom mieste skončil ACRM



(a) Reálne dáta (b) ACRM model (c) Random Waypoint

Obr. 4.3: *Empirická distribúcia celkového pokrytia.*

model. Ostatné modely boli viac ako niekoľkonásobne horšie (je to spôsobené aj tým, že v ostatných modeloch sme prešli väčšiu trajektóriu, a teda spravili väčšie pokrytie). V reálnych dátach sa stávalo, že sme monitorovali cestu aj keď sa mobilný uzol nehýbal (napr. cesta do školy, prednáška a cesta zo školy sa nám ukázala ako jedna cesta). V simulovaných dátach sa stále hýbeme (nerobíme pauzy), preto máme väčšie pokrytie ako reálne dáta. ACRM model ide cielene jedným smerom, pričom Levy Walk mení smer náhodne (smer vyberá z uniformnej distribúcie), preto sa Levy Walk vracia na miesta, kde už bol. V dôsledku toho má Levy Walk menšie pokrytie ako ACRM model, a teda distribúcia sa viac podobá na reálne dáta. Pre porovnanie prikladáme empirické distribúcie pokrytia pre reálne dáta (4.3a), ACRM model (4.3b) a Random Waypoint model (4.3c).

Záver

V práci sme sa venovali modelom pohybu - umelým algoritmom pre simuláciu pohybu mobilného uzla (MU). Zamerali sme sa na syntetické (náhodné) modely. V prvej časti našej práce sme popísali najčastejšie používané modely: Random Walk, Random Waypoint, Gauss-Markov, pravdepodobnostý Chiangov a Levy Walk model. Pre porovnanie modelov sme potrebovali príklady skutočného pohybu, preto sme si naprogramovali vlastnú aplikáciu na zbieranie GPS polohy, ktorú 42 dobrovoľníkov nosilo pri svojich každodenných cestách. Poukázali sme na technické problémy a ich možné riešenia, ktoré sa vyskytli pri vytvárani android aplikácie. Spolu dobrovoľníci nazbierali 948 „trackov[3]“, ktoré sme zjednodušili obdlžníkovou a uhlovou metódou. Cestu, ktorá sa skladala z krátkych ciest s malým vybočením, sme spojili do jednej väčšej cesty. Navrhli sme vlastnosti pohybu „jednotlivá cesta“, „podmienený uhol“ a „rozptyl“. Na každú vlastnosť sme pomocou Akaike informačného kritéria a MLE napasovali najlepšie sa hodiacu spojitú parametrickú distribúciu, ktorú sme využili pri návrhu nového modelu.

Náš ACRM „Aimed Constrained Random Movement“ model funguje na základe poznatkov zo skúmaných modelov v kapitole 1 a na základe odsledovaných vlastností reálneho pohybu v kapitole 3. MU si vyberie cieľ cesty z distribúcie podľa vlastnosti rozptyl a následne sa pohybujeme ku cieľu krátkymi cestami z distribúcie jednotlivá cesta, pričom vždy vyberieme uhol odklonenia od cieľa z distribúcie podľa vlastnosti podmienený uhol.

Najdôležitejšou časťou práce bolo navrhnutie metodiky porovnávania syntetických modelov a reálnych nazbieraných dát. Vybrali sme vlastnosti „trajektória“, „počet susedov“ a „celkové pokrytie“, vlastnosti dôležité z hľadiska prepojenia pohybu a efektivity routovacích protokolov v ad-hoc sieťach. Pre každú vlastnosť sme dostali empirickú distribúciu pre reálne aj simulované dáta. Následne sme vyhodnotili podobu empirických distribúcii reálnych dát s každým simulovaným modelom. Na porovnanie distribúcií sme využili dvojvýberový

[3]zoznam GPS súradníc a času pri presune z jedného miesta na druhé

Kolmogorov-Smirnov test. Táto metodika je prvým pokusom navzájom porovnať modely pohybu s reálnymi dátami a tak určiť ich úspešnosť. Analyzované vlastnosti sú prispôsobené nášmu cieľu (pohyb MU v ad-hoc sieti), preto v rôznych prípadoch sa môžu porovnávané vlastnosti upraviť podľa potreby. V našej analýze sme sa nezaoberali dôležitými vlastnosťami pohybu, ako sú pauzy a rýchlosť pre nedostatok v nazbieraných reálnych dátach (GPS aplikácia nám vždy zachytila len bod zmeny pohybu, a teda nevieme určiť ako dlho sme v niektorom bode čakali a ako dlho sme sa premiestňovali).

Ukázalo sa, že ACRM model dopadol podobne (pre vlastnosť počet susedov lepšie) ako ostatné modely. Výhoda ACRM modelu spočíva aj v tom, že malou obmenou parametrov (napr. obmedzením výberu cieľa, zmeny smeru, ...) vieme model prispôsobiť do konkrétnych situácií (podobne ako sledované modely). Vytvorili sme teda pružný model, ktorý v jednoduchej náhodnej forme dosahuje rovnako dobré, prípadne lepšie výsledky oproti ostatným syntetickým modelom a zároveň má potenciál na svoje vylepšovanie podľa potreby použitia pre konkrétne simulácie pohybu.

Literatúra

[1] Munjal, Aarti, Tracy Camp, and Nils Aschenbruck, 2012. *Changing Trends in Modeling Mobility.* Journal of Electrical and Computer Engineering, 2012.

[2] M. Musolesi, C. Mascolo, 2009. *Mobility Models for Systems Evaluation A Survey.* Book Chapter in Middleware for Network Eccentric and Mobile Applications. Benoît Garbinato, Hugo Miranda andd Luís Rodrigues Editors.

[3] Natarajan Meghanathan, 2010. *Impact of the Gauss-Markov Mobility Model on Network Conectivity, Lifetime and Hop Count of Routes for Mobile Ad hoc Networks.* Journal of Networks, Vol. 5, No. 5, May 2010

[4] T. Camp, J. Boleng, and V. Davies, 2002. . *A survey of mobility models for ad hoc network Research, Wireless Communication and Mobile Computing Special Issue on Mobile Ad Hoc Networking* . Research, Trends and Applications, 2(5):483–502, 2002.

[5] Vincent Gauthier, 2009. *Mobility model in ad-hoc network.* http://www-public.it-sudparis.eu/ gauthier/MobilityModel/mobilitymodel.html

[6] Johnson, David B., David A. Maltz, and Josh Broch, 2001.DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In Ad hoc networking 5 (2001): 139-172.

[7] Pelusi, Luciana, Andrea Passarella, and Marco Conti, 2006. *Opportunistic networking: data forwarding in disconnected mobile ad hoc networks.* In Communications Magazine, IEEE 44.11 (2006): 134-141.

[8] C. Chiang, 1998. *Wireless Network Multicasting.* PhD thesis, University of California, Los Angeles, 1998.

[9] *http://vyuka.compbio.fmph.uniba.sk/mbiwiki/index.php/Hľadanie_motívov,_cvičenia_pre_informatikov*

[10] D. Brockmann, L. Hufnagel, and T. Geisel, 2006. *The scaling laws of human travel.* Nature, vol. 439, pp. 462–465, Jan. 2006.

[11] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, 2008. *Understanding individual human mobility patterns.* Nature, vol. 453, pp. 779–782, Jun. 2008.

[12] Rhee, I., Shin, M., Hong, S., Lee, K., Kim, S. J., Chong, S., 2011. On the levy-walk nature of human mobility. IEEE/ACM Transactions on Networking (TON), 19(3), 630-643. 2011.

[13] *http://developer.android.com/reference/android/app/AlarmManager.html*

[14] *http://developer.android.com/reference/android/app/Service.html*

[15] *http://chrisrisner.com/Mobile-Geolocation-Apps-with-Windows-Azure-Websites-Part-4–Displaying-Points-of-Interest-in-Android*

[16] *http://developer.android.com/training/basics/firstapp/creating-project.html*

[17] *http://developer.android.com/reference/android/location/LocationManager.html*

[18] *http://developer.android.com/reference/android/content/BroadcastReceiver.html*

[19] *http://developer.android.com/reference/android/net/ConnectivityManager.html*

[20] *http://developer.android.com/reference/android/os/PowerManager.html*

[21] *http://developer.android.com/reference/android/os/PowerManager.WakeLock.html*

[22] *http://developer.android.com/reference/android/location/LocationListener.html*

[23] *http://www.jcraft.com/jsch/*

[24] *http://developer.android.com/reference/android/os/AsyncTask.html*

[25] Scholz, F. W., 1985. *Maximum likelihood estimation.* Encyclopedia of Statistical Sciences. 1985.

[26] Burnham, Kenneth P., and David R. Anderson, 2004. *Multimodel inference understanding AIC and BIC in model selection.* Sociological methods research 33.2: 261-304. 2004.

[27] *http://www.mathworks.com/help/stats/fitdist.html*

[28] *http://www.mathworks.com/matlabcentral/fileexchange/34943-fit-all-valid-parametric-probability-distributions-to-data/content/allfitdist.m*

[29] *http://www.physics.csbsju.edu/stats/KS-test.html*

[30] *http://www.mathworks.com/help/stats/kstest2.html*

[31] *http://radio.feld.cvut.cz/matlab/toolbox/stats/kstest2.html*

[32] *http://www.jsc.nildram.co.uk/*

Príloha A : DVD

Štruktúra DVD:

- GPS_aplikacia - apk súbor aplikácie a zdrojové kódy aplikácie
- GPS_nazbierane_data(surove) - nazbierané súbory od dobrovoľníkov (dáta zo servera)
- MATLAB - spísaný kód funkcie allfitdist
- realne_data(upravene) - výstup po upravení dát na samostanté „tracky“ (napr. 1_0_U.txt), „tracky“ upravené oblžníkovou metódou (napr. 1_0_U.txt) a „tracky“ následne upravené uhlovou metódou (napr. 1_0_A.txt)
- simulovane_data - výstupy simulácie rôznych modelov
- vlastnosti - výstupy zozbieraných vlastností pre reálne aj simulované dáta
- zdojove_kody - zdrojové kódy v jave ku úprave „trackov“ (rozbitie veľkého súboru dobrovoľníka na jednotlivé „tracky“ a použitie obdlžníkovej a uhlovej metódy), simulácii modelov a získavaniu vlastností pre analýzu