

UNIVERSITÀ DEGLI STUDI DI UDINE

Dipartimento di Scienze matematiche, informatiche e multimediali

Corso di Laurea Triennale in Informatica

Tesi di Laurea

**PRE-DESTINAZIONE:
MODELLI ED ESPERIMENTI PER
LA PREVISIONE DI TRAIETTORIE**

Relatore:
Ivan Scagnetto

Laureando:
ALESSIO PELLIN

ANNO ACCADEMICO 2014-2015

Introduzione

Negli ultimi anni, grazie all'avvento di nuove tecnologie nel settore delle reti wireless e dei dispositivi mobili le reti wireless sono diventate una parte fondamentale dell'infrastruttura delle telecomunicazioni. Ormai l'utente pretende che la qualità delle reti wireless siano almeno pari a quelli forniti via cavo visti gli incredibili passi avanti che le reti cellulari hanno ottenuto negli ultimi 5-6 anni. Basti pensare alla nuove reti 4G (LTE e WiMax) che promettono velocità nell'ordine sui 5-8 Mbps in download e contro le attuali linee ADSL con una media di 15 Mbps promessi. In più le attuali società telefoniche forniscono più facilmente reti wireless ad alta velocità piuttosto che reti cablate vista la diffusione di apparecchiature mobili quali smartphone, tablet, e così via [Figura 1]. In più con gli attuali progetti di Facebook e Google sulla progettazione di droni capaci di portare reti WiFi o LTE si punta ad avere una copertura quasi totale del suolo mondiale.

Questa sempre più crescente utenza connessa in movimento ha fatto accrescere l'interesse di acquisire informazioni contestuali per poter offrire servizi di maggior interesse e utilità per l'utente. L'informazione contestuale può riferirsi alla posizione dell'utente, all'ora attuale, a proprietà fisiche come la temperatura, ecc. La gestione efficiente del contesto richiede un dettagliato data modeling ottenuto con processi specifici di classificazione, inferenza e predizione.

Certo è difficile comprendere cosa può essere considerato context aware, senza prima avere dato una definizione di contesto. La principale definizione di contesto è la seguente: *Il contesto è qualsiasi informazione che può essere usata per caratterizzare la situazione di una entità. Un' entità è una persona, un luogo o un oggetto considerato rilevante all'interazione fra l'utente e l' applicativo, inclusi l'utente e l'applicativo stessi* [1]. Il contesto dunque è specificato mediante i valori di opportuni parametri, rappresentanti l'attività di un'entità. L'approccio orientato al contesto quindi permette all'entità di adattarsi all'ambiente, offrendo superiori vantaggi e possibilità ai nuovi applicativi.

Una delle proprietà che maggiormente si desidera nel mobile context-awareness è la possibilità di poter effettuare previsioni, caratteristica che permetterebbe lo sviluppo di nuove, avanzate applicazioni.



Figura 1: Copertura LTE mondiale

Nel 2001, il Computer Science and Telecommunications Board (CSTB) del Consiglio Nazionale della Ricerca degli Stati Uniti d'America ha riunito una commissione di esperti per eseguire una ricerca sulle opportunità certe e sui possibili sviluppi relativi all'interazione tra le comunità di ricerca geo-spaziale ed informatica. Nella relazione prodotta dalla commissione [2], si evidenzia come l'ubicazione dell'utente sia uno dei fattori fondamentali nelle diverse definizioni di contesto che sono state proposte in letteratura. La centralità di tale componente e la possibilità, fornita dalle attuali tecnologie mobili, di rilevare in modo sufficientemente preciso, semplice e continuativo, la posizione geografica degli utenti, fanno di questo settore un campo di ricerca attualmente molto attivo.

Contemporaneamente alla nascita dei servizi context aware, è dunque nata la necessità di avviare una ricerca mirata al miglioramento delle prestazioni nella previsione delle traiettorie. Tale ramo di ricerca ovviamente mutua gran parte della natura delle applicazioni context aware. In questo caso però le informazioni memorizzate si riducono a semplici dati spaziali e temporali, che possono ledere la privacy della persona in minima parte, ma non in modo così invasivo come, invece, possono essere portati a fare con applicativi context aware, che per loro natura cercano di tracciare tutti gli aspetti (gusti, personalità) dell'utenza. La conoscenza della posizione di oggetti mobili ha dunque condotto allo sviluppo di applicazioni e servizi che sono stati catalogati location-based, che necessitano di conoscere la posizione approssimata di un oggetto mobile

per operare. Esempio di tali servizi sono gli applicativi di navigazione, gestori di traffico e la pubblicità location-based. In uno scenario tipico, il dispositivo mobile che sta fornendo un determinato servizio, periodicamente informa il framework di posizionamento della attuale posizione.

Nell'ultimo decennio anche i sistemi di sistemi di posizionamento sono migliorati. da un'accuratezza di decine di metri, si è ormai arrivati ad avere una precisione di pochi metri anche per utilizzi civili. La ricerca di una precisione sempre più alta a disposizione di tutti ha portato alla creazione di sistemi di posizionamento alternativi al GPS quali GLONASS (Militare russo), COMPASS (Cina), Galileo (Europa) e IRNSS (India) stanno cercando di imporsi come alternativa al sistema Americano.

Grazie alla copertura sempre crescente di reti wireless e sistemi di posizionamento la context aware non è più incentrata sulla ricerca della posizione attuale dell'utente, ma sulle possibili destinazioni e percorsi nel breve e lungo termine. Grazie ad una metodologia si cerca di predire la posizione per anticipare i movimenti dell'utente e quindi cercare di eseguire un pre-fetch del servizio sulla località prevista. Lo sviluppo di pratiche ed accurate tecniche di previsione degli spostamenti può dunque aprire le porte a molti applicativi quali prenotazione di risorse, servizi location-based, ma anche migliorare la pianificazione e gestione delle aree urbane, grazie ad una migliore analisi del flusso urbano.

Ovviamente la possibilità di tracciare la posizione attuale dell'utente da sola non è sufficiente ad effettuare previsioni sul futuro dell'utente. In un articolo pubblicato qualche anno fa [3], nel quale si afferma come osservando i dati storici relativi ai movimenti di un utente (collezionati con diverse tecniche, nel caso citato accedendo alle basi di dati di una compagnia telefonica) sia possibile individuare pattern di movimento, e prevedere correttamente il luogo in cui si trova una persona per il 93% del tempo. Da tali studi emerge come ciascun individuo abbia un insieme di località che raramente lascia, e nelle quali si sposta con grande regolarità. Gli utenti che risiedono stabilmente in un raggio di 6 miglia hanno un tasso di predicibilità della posizione che va dal 93% al 97%, ma anche nel caso in cui il raggio aumenti di centinaia di miglia, la percentuale delle previsioni rimane alta, stabilizzandosi al 93% di successo. In tale articolo vengono inoltre indicati dei legami tra le località frequentate dall'individuo e le ore della giornata. In particolare si evince come in certi orari di transizione (prima o dopo l'orario di lavoro, oppure durante le pause pranzo) le previsioni sul luogo in cui si trova l'utente vedano un brusco peggioramento dei risultati; comunque la percentuale che l'utente si trovi in un qualsiasi momento della giornata nella località più visitata è molto alta (del 70%).

1 Obiettivi della tesi

Già in precedenti lavori di tesi svoltisi nell'Università degli Studi di Udine è stato presentato un algoritmo di previsione delle traiettorie basato sul modello della fisica dei campi elettrici, dove le località maggiormente importanti per l'individuo erano caratterizzate da forze attrattive e le meno importanti da forze nulle se non addirittura repulsive. Tale algoritmo è denominato ARDA, è stato presentato unitamente ad una nuova proposta di pesatura dell'importanza delle località che prende il nome di SpaceRank, il quale andava ad affiancare altri indici di valutazione delle località quali TotalTime (il tempo totale che l'individuo passa in una località), AverageTime (il tempo medio che l'individuo passa in una certa località durante una visita), NumberOfVisits (il numero di volte che l'utente passa per una certa località), e combinazioni di tali indici. Il lavoro svolto in una precedente tesi [3] proponeva:

- L'implementazione dell'algoritmo di previsione ARDA utilizzando vari indici di valutazione dell'importanza delle località, con diverse suddivisioni di territorio
- Il confronto dei risultati di previsione ottenuti evidenziando il comportamento dell'algoritmo ARDA rispetto a due altri algoritmi di previsione (il primo che suppone l'utente prosegua la propria corsa in linea retta mantenendo la velocità media, il secondo che suppone l'utente prosegua il proprio percorso rimanendo in un intorno del punto rilevato);
- La valutazione di stabilità degli algoritmi proposti eseguendo dei test anche su una diversa raccolta di punti GPS (messa a disposizione dal Prof. Thad Starner) riguardante un territorio di dimensioni e varietà di movimenti maggiori;
- L'analisi del comportamento dell'algoritmo nelle previsioni delle destinazioni finali, eseguite lungo quattro milestone virtuali poste lungo i percorsi (inizio, 25%, 50% e 75% del tracciato).

Questa tesi prende in esame l'algoritmo e i dati di quella precedentemente descritta e si pone i seguenti obiettivi:

- Traduzione degli algoritmi dal linguaggio Python a Java e miglioramento di alcuni aspetti;
- Unificazione delle procedure di test per i vari indici;
- Sviluppo di un'interfaccia per l'utilizzo al di fuori della riga di comando;
- Sviluppo di un sistema di visualizzazione ed esportazione dei test eseguiti.

2 Struttura della tesi

Lo scritto della tesi si divide in tre parti. La prima parte si occupa di presentare gli algoritmi presi in esame cercando di metterne in evidenza non solo gli aspetti modellistici, ma anche i problemi implementativi. Nella seconda parte si dà ampio spazio ai risultati ottenuti sia nei risultati che a livelli prestazionali. Infine la terza parte é quella conclusiva, in cui si riassume il lavoro svolto traendone le somme, presentando i punti di forza e di debolezza delle soluzioni adottate dando anche degli spunti di riflessione su eventuali sviluppi.

Le tre parti sopra esposte in realtà sono state svolte con diversa estensione, cercando di rispettare le gerarchie logiche. La prima sezione é racchiusa nei capitoli 1 e 2. Il primo capitolo espone i concetti e la teoria sugli algoritmi scelti (prendendo spunto dalla tesi [8]), vista l'importanza di riuscire a proporre nel modo più esauriente possibile le soluzioni algoritmiche adottate. Il secondo capitolo invece si sofferma sulle scelte di sviluppo fatte, gli accorgimenti fatti per migliorare gli algoritmi e il funzionamento del programma. La seconda parte invece si sviluppa nei seguenti due capitoli. Nel terzo capitolo si espongono i risultati ottenuti comparandoli con quelli della tesi precedente lasciando spazio a grafici e confronti cercando di capire il perché delle differenze. Nel quarto capitolo si fa una piccola osservazione sulle prestazioni delle 2 versioni facendo anche un confronto con i tempi stimati al tempo della tesi originale. Infine la terza parte, quella conclusiva, nella quale si cerca con il quinto capitolo di riassumere il lavoro svolto, evidenziando i risultati ottenuti e cercando di proporre come poter ulteriormente sviluppare la ricerca e migliorarne le prestazioni.

Indice

1	Obiettivi della tesi	vi
2	Struttura della tesi	vii
1	Teoria sugli algoritmi	1
1.1	Space Rank	1
1.1.1	Criteri di valutazione dell'importanza	1
1.1.2	PageRank	3
1.1.3	SpaceRank	5
1.2	ARDA	8
1.2.1	Formalizzazione del problema	9
1.2.2	Modello astratto di previsione	9
1.2.3	ARDA	12
2	Conversione e sviluppo	17
2.1	Analisi e scelte implementative	17
2.1.1	Conversione	17
2.1.2	Unificazione dei valori di default	18
2.1.3	Modifiche fatte	18
2.2	Interfaccia Java	19
2.2.1	Visualizzazione plot matrici	20
3	Comparative sui risultati ottenuti	21
3.1	Matrici d'importanza	21
3.1.1	Dati Udine	21
3.1.2	Dati Atlanta	21
3.2	ARDA	21
3.2.1	Dati Python	21
3.2.2	Dati Java	21
3.3	Valutazioni complessive	21

4	Comparative prestazionali	25
4.1	Confronto tra Java e Python	25
4.2	Confronto temporate	25
4.3	Test su dispositivi embedded	25
5	Conclusioni	27
5.1	Considerazioni	27
5.2	Sviluppi futuri	28
	Bibliografia	29

Capitolo 1

Teoria sugli algoritmi

1.1 Space Rank

Una buona conoscenza delle abitudini degli utenti, mediante i loro spostamenti, è di fondamentale importanza per poter sviluppare algoritmi di previsioni della destinazione. Infatti i luoghi di maggior interesse risulteranno essere le destinazioni dei loro spostamenti. Tuttavia, in questo particolare contesto, le definizioni del concetto di importanza e dei relativi indici di valutazione compongono un campo di ricerca di notevole interesse.

In questa sezione viene proposta una metodologia di analisi dei dati volta a stimare l'importanza delle località sulla base dei comportamenti degli utenti. Mediante i dati relativi inerenti alle visite effettuate degli utenti nelle singole località, ci si pone l'obiettivo di ottenere dei valori rappresentati l'importanza che ogni località assume nelle abitudini degli utenti.

1.1.1 Criteri di valutazione dell'importanza

Nella letteratura possiamo trovare diversi casi in cui risulta necessario lo sviluppo di sistemi di valutazione dell'importanza delle località come in in [4], nel quale viene presentato l'algoritmo **Predestination**, che esegue una suddivisione del territorio ed una valutazione dell'importanza di ciascuna sezione, per indicarne l'interesse per utente esaminato. Un altro esempio di particolare interesse è inoltre l'articolo [5], nel quale si propone un modello gravitazionale atto a considerare i flussi di una città per decidere in quali zone sia preferibile andare ad aprire nuove attività commerciali. Questi esempi ci fanno capire quanto sia importate la scelta di un appropriato criterio di valutazione di importanza delle regioni. Definire quali siano le località importanti per un utente non è tuttavia semplice. Alcuni studi [6] hanno dimostrato che le probabilità che ciascuno di noi si trovi in una certa località sono legate alla legge

di potenza, ovvero che tali probabilità relativamente ad una località sono proporzionali all'inverso della posizione occupata dalla località stessa in una scala di importanza. Questo ci fa dedurre che esiste un ordinamento delle località basato sull'importanza che le località hanno per ognuno di noi. Esaminando [?] possiamo inoltre dedurre, dato che le persone frequentano abitualmente gli stessi luoghi, che analizzando i comportamenti passati dell'utente possiamo definire un elenco ordinato di zone di interesse in modo tale da prevedere i suoi movimenti. Per definire l'importanza che ciascuna località ha per l'utente, e dunque intuitivo adottare i seguenti indici:

- $\#visits$: il numero di volte in cui l'utente è stato nella località;
- $avgTime$: il tempo medio delle visite effettuate dall'utente nella località;
- $totTime$: il tempo totale passato dall'utente nella località durante le sue visite.

Questi indici sono facilmente calcolabili da una qualsiasi collezione (previa standardizzazione della struttura del contenuto) ma, considerati singolarmente, possono dare solo una visione parziale delle abitudini degli utenti, e per alcune applicazioni non sarebbero sufficienti. Se valutiamo singolarmente gli indici notiamo che:

- $\#visits$ equipara le località di passaggio con quelle in cui avviene una lunga sosta;
- $avgTime$ equipara località frequentemente visitate a quelle che vengono raramente raggiunte;
- $totTime$ equipara sia le lunghe soste che l'alto numero di passaggio su una particolare cella.

Migliore dunque sembrerebbe l'utilizzo dell'indice $totTime$, in quanto esso riesce a valorizzare sia le lunghe permanenze che un alto numero di passaggi ma rimane però comunque il problema di come riuscire a differenziare le celle con stesso $totTime$. Per questo si è deciso di introdurre un indice derivato, definito come $combLinear$, composto dalla combinazione lineare degli indici $\#visits$ ed $totTime$. Questa soluzione dà la possibilità di differenziare le celle che hanno lo stesso valore ma porta il problema di come pesare i due indici durante il calcolo della combinazione lineare.

Dopo aver definito diverse proposte per la rilevazione dell'importanza delle località, ognuna con pregi e difetti, risulta interessante comparare, insieme agli

algoritmi di previsione di traiettorie basati sulle pesature delle località, un nuovo indice basato su una più complessa valutazione dei dati ma simile ai precedenti per risultati. Nella ricerca di una migliore valutazione delle località ci si è imbattuti nell'algoritmo di PageRank [7], utilizzato dai motori di ricerca per la catalogazione e l'ordinamento dei documenti ipertestuali. Tale algoritmo, mediante un grafo ottenuto dalla mappatura dei collegamenti ipertestuali tra pagine web, stima l'importanza dei documenti considerati. Si è dunque deciso di sviluppare il concetto proposto dall'algoritmo PageRank per sviluppare il nuovo indice *SpaceRank* che, mediante il grafo ottenuto dagli spostamenti dell'utente, stimerà l'importanza delle località di interesse dell'utente.

1.1.2 PageRank

L'algoritmo di PageRank [7] ha avuto origine nel 1995 da un progetto di ricerca della Stanford University, a cura di Larry Page. Quest'ultimo, insieme a Sergey Brin, svilupparono un motore di ricerca basato proprio sul questo algoritmo che negli anni a venire divenne il motore di ricerca più utilizzato, Google. Basato sull'idea di misurare l'importanza di ciascun elemento di un insieme di documenti con collegamenti ipertestuali mediante un valore numerico rappresentante la probabilità che una persona arrivi a quel dato documento scegliendo ciclicamente a caso uno dei collegamenti presenti nel documento ipertestuale che sta leggendo. Il processo matematico, utilizzato per il calcolo di tale valore per ciascuno dei documenti, si basa sulle catene di Markov. Prendendo in considerazione l'insieme dei documenti ipertestuali, viene calcolata la matrice $n \times n$, dove n è il numero dei documenti presenti nell'insieme. Su ciascuna riga vengono inseriti i valori relativi ai collegamenti ipertestuali che puntano a ciascuno dei documenti indicati dalle colonne. Sulla matrice ottenuta viene effettuata una normalizzazione tale da rendere la somma dei valori di ciascuna riga pari ad 1 ottenendo la matrice di adiacenza del grafo delle citazioni dell'insieme di documenti. La matrice risultante può essere interpretata come una catena di Markov ed il calcolo dell'autovettore dominante della matrice di adiacenza definisce la probabilità di trovarsi in ciascuno degli stati della catena di Markov durante una camminata casuale, ovvero la probabilità di raggiungere ciascuno dei documenti scegliendo ciclicamente a caso uno dei collegamenti ipertestuali contenuti nei documenti che vengono presentati, ovvero il valore di PageRank.

Tuttavia questo calcolo non è esente da problemi [8]. Il primo problema è relativo alla catena di Markov, la quale potrebbe non essere ergodica. Una catena di Markov si definisce ergodica se ciascuno stato della catena (ovvero ciascun nodo del grafo) è:

- aperiodico: se il suo periodo è pari a 1, ovvero se lo stato non deve sempre occorrere dopo un multiplo di un certo numero di passi;
- ricorrente positivo: se in una camminata casuale infinita c'è sempre probabilità di ritornare in uno stato già visitato in un numero finito di passi.

Il calcolo dell'autovettore dominante di una matrice relativa ad una catena di Markov non ergodica potrebbe non convergere e non si avrebbe, quindi, un limite stabile da poter utilizzare come valore di PageRank.

Il secondo problema, invece, riguarda proprio il concetto su cui si basa questa metodologia. È difficile pensare che utente sfogli un insieme di documenti seguendo solamente i collegamenti ipertestuali senza mai abbandonare un documento per uno qualsiasi dei documenti dell'insieme. Entrambi i problemi descritti possono essere risolti mediante l'utilizzo di quella che viene comunemente definita matrice di teletrasporto. Quest'ultima è una matrice $n \times n$ che definisce un grafo con un arco di pari peso tra ciascuna coppia di nodi. Tale grafo può essere interpretato come una catena di Markov in cui ad ogni passo è possibile passare dallo stato attuale ad uno qualsiasi degli stati della catena (compreso quello attuale). Mediante la combinazione lineare delle normalizzazioni delle due matrici descritte si ottiene la matrice il cui autovettore dominante contiene i valori di PageRank utilizzati come stima dell'importanza dei documenti dell'insieme. Per tale combinazione lineare viene utilizzato un fattore d , comunemente definito fattore di salto, che definisce la probabilità, stabilita a priori, che l'utente effettui un salto dal documento in cui si trova ad uno qualsiasi dei documenti dell'insieme, ovvero di seguire i collegamenti definiti dalla matrice di teletrasporto. Dunque, la matrice utilizzata per il calcolo dei valori di PageRank sarà:

$$P = (1 - d)xA + dxT \quad (1.1)$$

dove A è la matrice di adiacenza normalizzata del grafo delle citazioni, T è la matrice di teletrasporto normalizzata e d è il fattore di salto. Per il calcolo dell'importanza delle pagine web, dai motori di ricerca viene solitamente utilizzato un fattore di salto pari a 0.15.

La matrice di PageRank P ottenuta può essere utilizzata per calcolare la matrice di Importanza M_{imp} utilizzando il metodo delle potenze per ricavarne l'autovettore dominante. Sia x vettore qualsiasi di dimensione n , e P la matrice di PageRank otterremo la matrice M_{imp} come segue:

$$M_{imp} = \lim_{k \rightarrow +\infty} xP^k \quad (1.2)$$

La formulazione proposta in realtà non viene mai implementata in quanto presenta problemi di overflow ed underflow. Per evitarli è necessario eseguire ad ogni passo una normalizzazione del vettore ottenuto, costruendo una successione definita come segue:

$$\begin{cases} M_{aux}(i+1) = M_{imp}(i)P \\ \beta_{i+1} = \text{normaDue}[M_{aux}(i+1)] \\ M_{imp}(i+1) = \frac{M_{aux}(i+1)}{\beta_{i+1}} \end{cases} \quad (1.3)$$

che termina non appena $M_{imp}(n) - M_{imp}(n-1) < \varepsilon$ con ε scelto dall'utente, lasciando quindi $M_{imp} = M_{imp}(n)$.

1.1.3 SpaceRank

In questo capitolo viene proposta, dopo una breve introduzione concettuale, l'analisi del comportamento dell'algoritmo SpaceRank [8] che, come l'algoritmo di PageRank da cui trae ispirazione, ha come obiettivo la ricerca dei valori di importanza delle località, calcolate mediante l'autovettore delle matrice ottenuta dall'agoritmo.

Formalizzazione del problema

L'obbiettivo di questo algoritmo è, dato un insieme di informazioni relative all'ubicazione di un insieme di utenti in una data area e lungo un dato periodo di tempo, di stimare l'importanza delle località dell'area di interesse basate sulle informazione dell'insieme. Queste località di interesse sono un insieme di sotto-aree $L = l_0, l_1, \dots, l_{n-1}$ ottenute della divisione dell'area di interesse totale. Tali località potrebbero avere forme e dimensioni differenti, ma per semplicità, in questa discussione supporremo che abbiamo forma e dimensioni identiche. In Figura 1.1 sono rappresentati alcuni esempi grafici delle località sopra descritti.

Definiamo N_i come l'insieme di tutte le località vicine ad l_i , ovvero l'insieme di tutte le località raggiungibili da l_i con un solo passaggio, compresa l_i stessa, ed N come l'insieme di tutti gli insiemi N_i . Disponendo dei dati relativi agli spostamenti, quindi dell'ubicazione e della velocità, di uno o più utenti durante un periodo di tempo possiamo, per ciascun utente, creare un registro storico $R_u = r_0, r_1, r_2, \dots$. Ogni elemento dell'insieme è una tripla $r_i = (t, l, s)$ dove t è un riferimento temporale relativo all'istante in cui è stata effettuata la campionatura, $l \in L$ è la località registrata ed s è la velocità di spostamento dell'utente in quell'istante. Definiamo quindi $R = R_0, R_1, \dots, R_n$ l'insieme dei registri relativi agli n utenti del campione. Dati gli insiemi L, N ed R ,

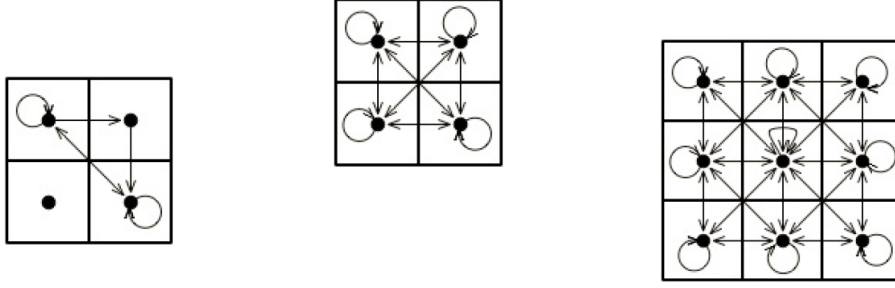


Figura 1.1: Esempi di grafi delle località

l'obiettivo è quello di ottenere un valore numerico che stimi l'importanza di ciascuna località $l \in L$.

Matrice basata sul registro storico dei dati

Come prima procedura, l'algoritmo SpaceRank prevede la mappatura dei registri relativi alla campionatura dell'ubicazione degli utenti in un grafo pesato diretto, rappresentante le abitudini degli utenti osservati. Sia A la matrice di adiacenza $n \times n$ (dove n è la cardinalità di L) del grafo delle abitudini, inizialmente nulla (cioè $A[i, j] = 0$ per ogni i e j compresi tra 0 ed $n-1$). Dato l'insieme R , per ciascun $R_u \in R$, per ciascuna coppia di elementi di R_u temporalmente successivi r_x ed r_{x+1} , deve essere eseguito l'aggiornamento $A[i, j] = A[i, j] + 1$, dove l_i è la località indicata dalla tripla r_x ed l_j è la località indicata dalla tripla r_{x+1} . Dopo aver mappato tutti gli spostamenti, ogni elemento $A[i, j]$ della matrice A conterrà il numero di movimenti (ovvero passaggi da una località ad un'altra) dalla località i alla località j effettuati dagli utenti durante il periodo di osservazione. È da notare che se l'utente non compie uno spostamento di località durante 2 campionamenti temporalmente successivi, questo spostamento verrà mappato come un ciclo sullo stesso nodo del grafo. Definiremo questo spostamento come permanenza.

Quanto descritto appena descritto è visibile mediante di grafici in Figura 2.1. Vengono visualizzate le divisione in località di un'area di interesse con relativi archi (nella figura non sono visibili i pesi assegnati agli archi) rappresentanti gli spostamenti. Per ogni immagine di può notare che nella località in alto a sinistra si può vedere un caso di permanenza. Tramite un processo di normalizzazione delle righe della matrice A è possibile ottenere una matrice \bar{A} , in cui la somma dei valori di una singola riga sia 1. Ciascun valore $\bar{A}[i, j]$ contenuto in tale matrice può essere interpretato come la probabilità che l'utente effettui un movimento dalla località i alla località j sulla base delle abitudini osservate,

contenute nell'insieme dei registri R . Per normalizzare i valori si può dividere ciascun elemento di ciascuna riga della matrice per la somma dei valori della riga a cui esso appartiene. Questo è uno dei molti procedimenti possibili, ma anche uno dei più veloci.

Matrice di transizione

La matrice \bar{A} finora ottenuta non è utilizzabile per ottenere una soluzione corretta in quanto si avrebbe una probabilità pari a zero di raggiungere una località mai visitata prima dall'utente e la catena di Markov relativa al grafo delle abitudini potrebbe non essere ergodica. A tal fine può essere creata una matrice \bar{B} , che definiremo come matrice di transizione, che rappresenti un grafo avente un nodo per ciascuna località $l \in L$ ed un arco per ciascuna coppia di località contigue. Sia dunque \bar{B} la matrice definita come:

$$\bar{B}[i, j] = \begin{cases} \frac{1}{c_i} & \text{se } j \in N_i \\ 0 & \text{altrimenti} \end{cases} \quad (1.4)$$

dove c_i è la cardinalità dell'insieme N_i . Tale matrice è già normalizzata e descrive un grafo corrispondente ad una catena di Markov ergodica in cui si ha la stessa probabilità di effettuare ad ogni passo una transizione dallo stato attuale ad uno qualsiasi degli stati che rappresentano le località vicine a quella in cui si trova l'utente (compresa quest'ultima). Ad esempio in Figura 2.1 (b) viene illustrato il grafo di una possibile matrice \bar{B} relativa ad una area suddivisa in quattro località disposte in una griglia 2×2 (nella figura non sono visibili i pesi assegnati agli archi). In questo caso particolare, l'insieme dei vicini di ciascuna località comprende tutte le località dell'area di interesse. Lo stesso vale per la località centrale in Figura 2.1 (c), dove viene illustrato il grafo di una possibile matrice \bar{B} relativa ad un area suddivisa in nove località disposte in una griglia 3×3 (nella figura non sono visibili i pesi assegnati agli archi), ma non per le altre località di questo secondo esempio, dalle quali è possibile raggiungere solo alcune località (4 o 6 a seconda della posizione).

Combinazioni delle matrici

Dopo aver ottenuto le due matrici \bar{A} e \bar{B} rappresentanti rispettivamente le abitudini osservate degli utenti e la conformazione dell'area d'interesse, possiamo combinare le due matrici per ottenere una matrice normalizzata a cui applicare il calcolo dell'autovettore dominante. Nel calcolo del PageRank viene definito il fattore di combinazione d che, in questo contesto, possiamo definire come fattore di comportamento non abitudinario, dato che definisce la percentuale di probabilità con cui deve essere presa in considerazione la possibilità

che l'utente si muova al di fuori delle abitudini osservate. La matrice a cui verrà dunque applicato il calcolo dell'autovettore dominante sarà:

$$S = (1 - d)x\bar{A} + dx\bar{B}$$

Come avviene nel calcolo del PageRank, viene impostato $d = 0.15$. Un valore minore darebbe maggiore importanza alle abitudini osservate, mentre un valore superiore porterebbe in considerazione un comportamento generico e non legato alle abitudini osservate. I due casi estremi porterebbero a $S = \bar{A}$, con $d = 0$, ed a $S = \bar{B}$, con $d = 1$. Ovviamente questi ultimi due casi sono da escludere: il primo perché porterebbe agli stessi problemi a causa dei quali è stata introdotta nel calcolo la matrice \bar{B} ; il secondo perché non darebbe alcuna effettiva informazione relativa ai comportamenti osservati degli utenti.

Calcolo della Matrice di Importanza

Una volta calcolata la matrice di SpaceRank S , non rimane che utilizzarla per calcolare la matrice di Importanza M_{imp} utilizzando il metodo delle potenze per ricavare l'autovettore dominante.

Sia x vettore qualsiasi di dimensione n , ed S la matrice di SpaceRank otterremo la matrice M_{imp} come segue:

$$M_{imp} = \lim_{k \rightarrow +\infty} xS^k \quad (1.5)$$

Ovviamente per calcolare effettivamente M_{imp} sarà sufficiente eseguire la seguente successione libera da problemi di underflow ed overflow:

$$\begin{cases} M_{aux}(i+1) = M_{imp}(i)S \\ \beta_{i+1} = \text{normaDue}[M_{aux}(i+1)] \\ M_{imp}(i+1) = \frac{M_{aux}(i+1)}{\beta_{i+1}} \end{cases} \quad (1.6)$$

che termina non appena $M_{imp}(n) - M_{imp}(n-1) < \varepsilon$ con ε scelto dall'utente, lasciando quindi $M_{imp} = M_{imp}(n)$.

1.2 ARDA

Come esposto nell'introduzione, negli ultimi anni sono stati sviluppati diversi algoritmi per la previsione delle traiettorie degli utenti basati su complicati modelli matematici e statistici basati su informazioni raccolte nel passato. In questo capitolo, dopo una breve formalizzazione del problema della previsione delle traiettorie, viene proposto un modello astratto presentato in [8] basato sull'utilizzo di un semplice modello fisico dei campi elettrici.

1.2.1 Formalizzazione del problema

Si prenda ad esempio un generico oggetto in movimento su una superficie bidimensionale. Possiamo indicare tale oggetto generico come oggetto della previsione (ODP). L'obiettivo è quello di determinare la probabilità con cui l'ODP, che all'istante t_0 è posizionato nel punto (x_0, y_0) sulla superficie, si potrà trovare in ciascuno dei punti del piano ad un certo istante t_i nel futuro, dato un suo attuale vettore di spostamento \vec{v}_0 . In prima istanza, questo equivale a cercare una funzione a sette parametri:

$$p(x_0, y_0, t_0, \vec{v}_0, t_i, x, y) \quad (1.7)$$

che calcoli un valore di probabilità per ciascun punto (x, y) del piano. Come già fatto notare, gli spostamenti delle persone non sono puramente casuali, ma dettati dall'interesse di esse verso un piccolo insieme di luoghi [6]. Proprio questi luoghi possono esercitare un'attrazione dell'ODP nei suoi spostamenti visto l'influenza che essi hanno nelle storico fornito dell'utente. Da questo possiamo ricavare una funzione del tipo:

$$p[k_1, \dots, k_n](x_0, y_0, t_0, \vec{v}_0, t_i, x, y) \quad (1.8)$$

che calcoli la probabilità che (x, y) sia la posizione dell'ODP al tempo t_i , dato uno schema fissato di punti di interesse sulla superficie $[k_1, \dots, k_n]$.

1.2.2 Modello astratto di previsione

Le metafore legate al mondo reale sono largamente utilizzate in diversi campi dell'informatica. In questo capitolo viene presentato un modello astratto, per la previsione delle traiettorie degli utenti sfruttando una metafora legata alla fisica dei campi elettrici.

Modello gravitazionale

Un recente studio [6] ha confermato che qualunque persona è legata ad un insieme di luoghi che possiamo definire importanti. Le analisi condotte indicano che vi sono alte probabilità che in un qualsiasi momento una data persona si trovi in due o tre località importanti e basse probabilità che si trovi in un qualsiasi altro posto. Da queste considerazioni si può dedurre che se un utente è in movimento ci sono alte probabilità che si stia dirigendo verso uno di questi punti di interesse. Infatti un gran numero di persone compiono gli stessi tragitti per arrivare a delle località importanti (sia per abitudine o per necessità). Tuttavia, durante la giornata effettuiamo anche altri spostamenti

che ci portano in località non appartenenti a quel piccolo insieme definite importanti. In questi casi la probabilità che la destinazione sia un nostro percorso sia una di queste località meno importanti è bassa, ma non nulla.

Anche la velocità e la direzione di uno spostamento può influenzare il risultato della previsione, perché, ad esempio, se un utente procede lentamente ci saranno più probabilità che si fermi in una località vicina a quella attuale, anche se meno importante, invece che in una più lontana ma con maggiore importanza. Se invece l'utente procede a velocità sostenuta ed in direzione tangenziale rispetto ad una località non molto importante, la probabilità che si assegnerebbe intuitivamente a quella località sarebbe bassa.

Supponiamo quindi di conoscere quali sono le località importanti per un dato utente, ovvero di utilizzare una scala di importanza con un valore assegnato a ciascuna di queste e di conoscere la disposizione spaziale di tali località in un piano bidimensionale che rappresenti l'area di interesse in cui l'utente si muove. Supponiamo inoltre di conoscere i dati relativi alla velocità ed alla direzione di movimento dell'utente in un certo istante temporale t_0 . Date tali informazioni è possibile creare un modello fisico gravitazionale che rappresenti la situazione in cui si trova l'utente all'istante t_0 secondo il modello astratto precedentemente descritto. Si parte da un modello fisico di spazio tridimensionale vuoto e si aggiunge, su uno stesso piano nello spazio tridimensionale, un oggetto fisso per ciascuna delle località importanti secondo la loro disposizione nell'area di interesse, assegnando a ciascun oggetto una massa proporzionale all'importanza della località che rappresenta. La presenza di questi oggetti fissi nello spazio tridimensionale darà origine ad uno schema di forze quale risultato dell'interazione dei campi di attrazione gravitazionale generati dai diversi oggetti. Aggiungendo al modello fisico un oggetto puntiforme di massa trascurabile, il cui vettore di spostamento sul piano su cui sono stati disposti gli oggetti fissi corrisponde alla velocità ed alla direzione dell'utente all'istante t_0 , il movimento di tale oggetto dipenderà dalle forze di attrazione gravitazionale generate dagli altri corpi presenti (vedi Figura 1.2).

Generazione del modello

La metafora legata alla fisica gravitazionale è probabilmente quella più intuitiva per questo modello astratto di previsione della traiettoria. Da notare che anche altre leggi fisiche, quali la diffusione del suono e i campi magnetici, hanno modelli fisici simili a quelli gravitazionali. Questi ultimi infatti possono essere particolarmente interessanti poiché prevedono la presenza sia di potenziali negativi che esercitano forze attrattive (per oggetti di carica positiva) come quelle dei campi gravitazionali, sia di potenziali positivi generati

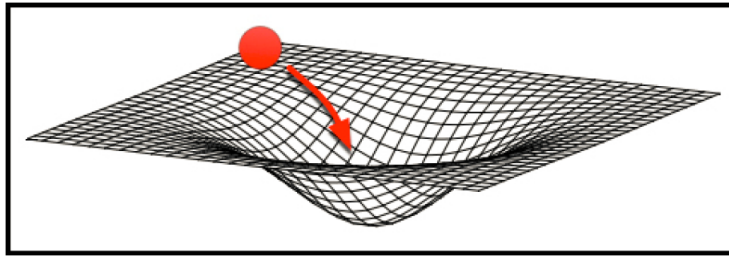


Figura 1.2: Esempio di attrazione gravitazionale

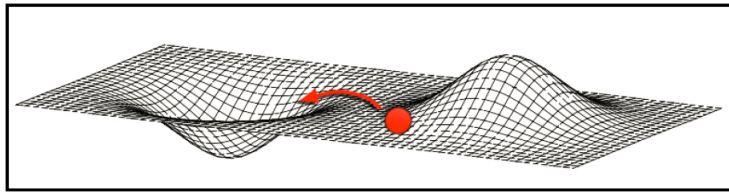


Figura 1.3: Arda: Esempio di attrazione e repulsione gravitazionale

dalla presenza di cariche positive che esercitano forze repulsive (per oggetti di carica positiva), come illustrato in Figura 1.3.

L'introduzione di forze repulsive nel nostro modello può sembrare inutile, ma analizzando bene il concetto di repulsività possiamo fruttarlo in casi particolari. Un esempio può essere l'indicare zone che non posso essere raggiunte oppure oltrepassate per via di limiti fisici o strutturali come ad esempio il passaggio in mare o attraverso fiumi, montagne o edifici. In questi casi la traiettoria dovrà essere deviata per non incorrere in tragitti fisicamente impossibili e/o non veritieri. Ci sono anche casi più sottili, come divieti di transito o zone ad accesso limitato, che, seppur non limitate per ostruzioni naturali o fisiche, rimangono sempre zone da evitare in un possibile calcolo del destinazione.

Come si può notare, in questo tipo di modello fisico, basato sui campi elettrici, è più semplice ragionare in termini di potenziale presente in una certa località piuttosto che di presenza di una certa carica nella località stessa. Inoltre, l'analisi delle abitudini di un utente darà come risultato un valore di importanza da assegnare a ciascuna delle località prese in considerazione e potrebbe non essere banale scegliere un piccolo insieme di punti di interesse disposti sulla superficie. Si può quindi scegliere di interpretare i valori di importanza, assegnati a ciascuna località, come i valori opposti del potenziale elettrico presente in quella data località. In tal modo, maggiore è l'importanza, minore è il valore del potenziale e quindi maggiore è l'attrazione esercitata dalla località.

1.2.3 ARDA

Dopo aver fatto le dovute premesse, possiamo descrivere l'algoritmo di previsione basato sul modello astratto precedentemente descritto nel paragrafo 1.2.2. In questo paragrafo vengono introdotte le formule utilizzate per eseguire una previsione della traiettoria dell'utente, sia a breve termine che a lungo termine, sulla base dei valori di importanza assegnati alle località secondo un dato indice di stima. Le simulazioni fisiche descritte vengono effettuate in uno spazio bidimensionale a valori continui rappresentante l'area di interesse. Poiché i dati ottenuti dall'analisi degli spostamenti degli utenti sono relativi ad un insieme discreto di località, il piano utilizzato nella simulazione deve essere suddiviso in sezione di ugual dimensione. Durante la simulazione, l'applicazione delle forze in gioco previste dal modello astratto viene operata in modo omogeneo a tutti i punti del piano continuo appartenenti ad una stessa sezione della griglia, ovvero a tutti i punti dello spazio appartenenti ad una stessa località. Si suppone che i valori di importanza utilizzati siano disponibili per mezzo di una struttura dati di tipo matriciale M_{imp} che rappresenti la griglia delle località.

Utilizzo dei valori di importanza

Utilizzando un modello gravitazionale, possiamo definire una matrice M_{pot} che rappresenti l'energia potenziale elettrica di ciascuna località. Ciascun ODP sarà attratto dalle aree a basso potenziale, associate con le località più importanti per l'individuo. Per definire i valori di M_{pot} è possibile utilizzare qualsiasi indice di località già indicati in precedenza:

- $\#visits$: il numero di volte in cui l'utente è stato nella località;
- $avgTime$: il tempo medio delle visite effettuate dall'utente nella località;
- $totTime$: il tempo totale passato dall'utente nella località durante le sue visite;
- $spaceRank$: basato su un'analisi dello storico degli spostamenti dell'utente e sull'analisi dell'intorno di ciascuna località.

Al di là degli indici adottati per l'analisi del territorio, importante e comprendere come il risultato di tale procedura sia una matrice di importanza M_{imp} , che associa i valori più alti alle località dalla maggiore importanza. Dato che il nostro modello fisico utilizza valori di importanza opposti, dobbiamo generare partendo da M_{imp} la matrice a valori opposti M_{pot} , rappresentante

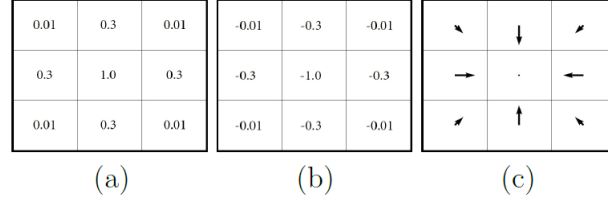


Figura 1.4: (a) Valori di importanza assegnati alle località M_{imp} ; (b) valori di potenziale assegnati alle località M_{pot} ; (c) vettori di accelerazione M_{acc}

il potenziale delle località. Per ottenere ciò si è scelto di utilizzare la funzione $f(x) = -x$, ma sarebbe stato possibile adottare anche funzioni di tipo polinomiale, logaritmico od esponenziale. A questo punto non rimane altro che calcolare la matrice delle accelerazioni M_{acc} dipendenti dalla matrice M_{pot} appena calcolata. In particolare ogni cella conterrà il vettore accelerazione relativo alla località rappresentata dalla cella della matrice. M_{acc} è dunque calcolato utilizzando il gradiente:

$$M_{acc} = -\vec{\nabla}(M_{pot}) \quad (1.9)$$

Come ultimo passaggio andiamo a normalizzare i vettori di accelerazione ottenuti in M_{acc} in modo che la media delle accelerazioni rilevate nei dati acquisiti corrisponda alla media delle accelerazioni contenute in M_{acc} . Per fare ciò è opportuno seguire la seguente procedura:

1. Rilevare tutte le magnitudo dei vettori di accelerazione dai dati acquisiti e calcolarne il valore medio: $AvgRealMagnitude$
2. Rilevare tutte le magnitudo dei vettori di accelerazione di M_{acc} e calcolarne il valore medio: $AvgMaccMagnitude$
3. Per ogni vettore accelerazione di M_{acc} moltiplicare gli elementi del vettore per la costante ricavata da $\frac{AvgRealMagnitude}{AvgMaccMagnitude}$

Calcolo della traiettoria futura

Dopo aver calcolato e impostato l'importanza per ogni località, si può procedere al calcolo della previsione della traiettoria utilizzando:

- Un tempo di campionamento, indicante la profondità temporale in secondi della singola previsione. Nelle nostre rilevazioni abbiamo utilizzato $\Delta_t = 1$

- La posizione $(x_1; y_1)$ al tempo corrente t_1
- La posizione $(x_0; y_0)$ al tempo $t_0 = t_1 - \Delta_t$
- La matrice di accelerazione M_{acc} , dove $M_{acc}[x, y]$ contiene il vettore di accelerazione per la località (x, y)
- Una funzione frenante $\gamma(M_{imp}, x, y)$, che ritorni il fattore frenante in base all'importanza della località

Quest'ultima funzione viene proposta per migliorare l'iterazione tra l'ODP e le forze attrattive delle località di maggior interesse. Infatti le località poco importanti o con forza repulsiva giustamente respingono l'ODP in modo non passi in quelle zone, ma le località di maggior interesse devono avere un'ulteriore forza che permetta all'ODP di rallentare l'avanzamento fino anche a fermarlo. Nel nostro caso abbiamo adottato una funzione frenante che garantisca fattori frenanti che andassero dal 1.5% al 10% del valore ad ogni previsione, a seconda dell'importanza della località percorsa. Tale decisione ha dunque portato alla definizione della seguente funzione:

$$\gamma(M_{imp}, x, y) = 0.015 + 0.085 \times M_{imp}[x, y] \quad (1.10)$$

Una volta definiti i vari elementi del processo previsionale, non rimane che proporre la formula adottata per simulare l'itinerario dell'ODP e quindi per prevedere la posizione (x_2, y_2) dell'utente al tempo $t_2 = t_1 + \Delta_t$. Tale funzione è la Verlet integration formula:

$$\begin{aligned} (x_2; y_2) = & \{ [2 - \gamma(M_{imp}, x_1, y_1)] \times (x_1, y_1) \} \\ & - \{ [1 - \gamma(M_{imp}, x_1, y_1)] \times (x_0, y_0) \} \\ & + \{ M_{acc}[x, y] \times \Delta_t^2 \} \end{aligned} \quad (1.11)$$

Tale formula può essere applicata iterativamente per calcolare le future posizioni dell'ODP in ogni tempo $t_k = t_1 + (k - 1) \times \Delta_t$, con $k \geq 2$.

Utilizzo di informazioni aggiuntive

L'algoritmo descritto prevede l'utilizzo della matrice M_{imp} costruita in base ai valori di importanza ottenuti dall'analisi del comportamento dell'utente durante il periodo di osservazione. Tuttavia, il modello astratto prevede l'utilizzo di informazioni sull'importanza delle località indipendentemente dalla fonte di tali dati.

Come suggerito dalla precedente tesi, per poter migliorare la previsione delle traiettorie si potrebbero valutare anche altri valori al di fuori degli indici di

importanza. Uno su tutti potrebbe essere l'arco temporale in cui svolgo la simulazione. Eseguire la simulazione di uno spostamento durante la settimana potrebbe essere abbastanza preciso visto che quasi tutte le persone lavorano in un luogo fisso e vi restano in una determinata fascia oraria. Lo stesso test eseguito in uno spostamento nel weekend darebbe risultati sfalsati perchè si avrebbe il luogo di lavoro come località di interesse. Dividere le simulazioni per giorno della settimana potrebbe comportare un notevole miglioramento sia dell'analisi dell'importanza delle località, sia del calcolo della destinazione. Allo stesso modo è potremmo utilizzare altre tipologie di dati non strettamente legate agli spostamenti ma che possono essere integrate in questo algoritmo e sfruttate in alcune particolari situazioni come gli impegni in un calendario (sfruttando l'orario e il luogo dell'appuntamento) oppure da reti sociali (controllando il posizionamento di amici, colleghi o eventi a cui abbiamo aderito). La flessibilità del modello nella codifica dei dati da utilizzare come informazioni sull'importanza delle località viene sottolineata dalla semplicità con la quale le diverse tipologie di dati possono essere integrate in un'unica valutazione. Due o più fonti di dati possono infatti essere sovrapposte, mediante una combinazione lineare delle matrici di importanza relative alle singole fonti, generando un'unica matrice:

$$M_{imp} = \alpha_1 M_{fonte1} + \alpha_2 M_{fonte2} + \dots \quad (1.12)$$

dove $\alpha_1 M_{fonte1} + \alpha_2 M_{fonte2} + \dots$ sono le matrici relative alle singole fonti ed $\alpha_1, \alpha_2, \dots$ sono i rispettivi pesi utilizzati per la combinazione lineare.

Capitolo 2

Conversione e sviluppo

2.1 Analisi e scelte implementative

Java e Python sono due linguaggi di programmazione simili ma diversi tra loro nati e progettati per necessità differenti. Java nato agli inizi degli anni '90, è uno dei linguaggi di programmazione più conosciuti ed apprezzati nel decennio passato. Ha avuto un notevole successo grazie all'esposizione di internet e la possibilità di creare applicazioni client-server indipendenti dalla piattaforma. Simile al C++ come sintassi di base ma senza alcune sue caratteristiche che potevano creare delle criticità a livello di progettazione (come l'aritmetica dei puntatori, ecc)[9], mentre le caratteristiche object oriented sia a C++ ma soprattutto all'Objective C.

Il codice Java deve essere prima compilato per ottenere il cosiddetto *bytecode* che poi verrà eseguito attraverso una fase di interpretazione ad opera della Java Virtual Machine. Proprio questo procedimento riesce a slegare i programmi dall'ambiente di compilazione ed esecuzione (da cui il motto *Write once, Run everywhere*).

Python, nato sempre agli inizi degli anni '90, è un un linguaggio multi-paradigma, che fa della dinamicità, semplicità e flessibilità i suoi principali obiettivi. Supporta la programmazione Object oriented e molte caratteristiche di programmazione funzionale e di riflessione. Una sua caratteristica che lo contraddistingue è la non tipizzazione delle variabili.

2.1.1 Conversione

Sono proprio le differenze appena descritte che si sono contraddistinte durante la conversione del progetto da Python a Java. La struttura più rigida di Java ha richiesto un'analisi delle variabili, delle funzioni e degli oggetti uti-

lizzati che, a volte, ha richiesto una conversione della tipologia delle variabili oppure una divisione per ogni tipologia utilizzata dal programma. Sebbene questa rigidità abbia richiesto un'analisi preventiva del codice, in seguito ha reso lo sviluppo più agevole in quanto il flusso dei dati rimane coerente visto che non può variare di tipologia.

2.1.2 Unificazione dei valori di default

Nel progetto originale si era preferito dividere sia l'esecuzione dei test tra il percorso di Udine con quello ad Atlanta, sia il calcolo delle matrici di importanza con l'algoritmo SpaceRank dagli altri metodi, questo perché si è preferito concentrare lo sviluppo degli algoritmi e dei test per i casi specifici. Tale approccio ha portato alla realizzazione di quattro distinti flussi di codice simili tra loro ma con variabili e funzioni indipendenti e non riutilizzabili. Durante l'analisi iniziale, necessaria per definire funzioni e variabili, si è preferito unificare le procedure simili in modo da semplificare e diminuire il codice necessario per il programma. Anche le strutture dati sono risultate simili optando per unificare la gestione del calcolo della matrice di importanza.

2.1.3 Modifiche fatte

Durante lo sviluppo dell'applicativo si sono voluti apportare dei cambiamenti a livello di calcolo.

Calcolo della distanza

Nella versione precedente, per calcolare le distanze tra due punti successivi, ottenuti dalle rilevazioni, veniva usato il calcolo della distanza lineare tra 2 punti. Nel nostro caso non possiamo considerare i nostri spostamenti su un piano ma spostamenti su una superficie sferica. Per questo si è preferito sostituire

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (2.1)$$

con il calcolo della distanza su una superficie sferica

$$d = r_{earth} * \arccos((\sin lat_1 * \sin lat_2) + (\cos lat_1 * \cos lat_2 * \cos(lat_1 - lat_2))) \quad (2.2)$$

Questa variazione rende la distanza calcolata più vicina alla realtà, soprattutto quando i due punti sono molto lontani tra loro, in alcuni casi limite, può ancora non essere precisa.

Negli spostamenti presi in esame questo cambiamento porta un miglioramento

marginale ma si è preferito introdurla soprattutto per avere una base di sviluppo più accurata.

Calcolo della matrice accelerazione

Nel calcolo delle matrici ARDA, per ottenere la matrice delle accelerazione M_{acc} si esegue, sulla matrice M_{imp} , una funzione gradiente per calcolare i valori del vettore in tutte le zone. Come si può vedere in FIGURA ???, la matrice generata ha ottimi risultati con punti di importanza distanti tra loro ma con punti vicini il calcolo del vettore ha qualche incongruenza nelle zone adiacenti. Utilizzando l'algoritmo di Sobel, questo distribuisce con maggior equità le forze anche in presenza di molti punti di interessi vicini tra loro.

Sia nel precedente progetto che in questo, la matrice di accelerazione viene calcolata dopo un solo processo della matrice M_{imp} calcolando le forze solo nelle zone adiacenti a quelle con dei valori di interesse. Pensando proprio all'idea su cui si basa l'utilizzo delle matrici di accelerazione si può capire quanto possa essere limitativo eseguire questo calcolo solo in queste zone. Questo può portare ad un ulteriore sviluppo e confronto con il calcolo della matrice di accelerazione con più ricalcoli sulla matrice risultante

$$M_{acc} = G^n(M_{imp}) \quad (2.3)$$

2.2 Interfaccia Java

Un'ulteriore obiettivo del progetto è creare un'interfaccia user-friendly per poter eseguire i setup e visualizzare i risultati in modo semplice e veloce. Nel progetto precedente bisognava eseguire i setup dei file, directory e impostazioni direttamente nei vari file del codice.

FIGURA ???

Adesso l'utente ha la possibilità di impostare:

1. i file mediante una semplice selezione tramite un finder navigando nel filesystem
2. selezionare le dimensione delle celle e gli indici utilizzati per il calcolo
3. visualizzare un plot della matrice di importanza.

Per l'esecuzione dei test si è scelto di dividere il calcolo dal salvataggio dei risultati mediante i file CSV già implementati nella versione precedente, con

la possibilità di visualizzare solamente il dato preso in esame. Proprio per rendere facilmente utilizzabile la funzione di visualizzazione dei dati, la sezione dei grafici è divisa in varie schede (come si può vedere dall'immagine) generate in base alle opzioni selezionate.

2.2.1 Visualizzazione plot matrici

Questa funzionalità ha però tolto la possibilità di selezione multipla della grandezza delle zone di importanza e impostato un limite di 2 possibili tipi di indice.

FIGURA ??

Infatti la visualizzazione e la gestione delle schede sarebbe stata troppo dispersiva data la possibilità di generare 25 casi separati con almeno 7 grafici differenti per ognuno di essi ed un totale di 175 schede massime. Con le limitazioni imposte si arriva a un massimo di 14 schede divise in 2 righe, sufficienti per avere subito dei grafici e poter fare un confronto veloce tra 2 tipi di indice differenti.

FIGURA ??

Questa piccola variazione però non influenza la generazione dei file CSV con i risultati. Essi non vengono cancellati e, con il lancio di tutte le procedure mediante l'interfaccia, possono essere facilmente reperibili per effettuare analisi direttamente dai dati che normalmente si visualizzano nei grafici.

Capitolo 3

Comparative sui risultati ottenuti

3.1 Matrici d'importanza

3.1.1 Dati Udine

3.1.2 Dati Atlanta

3.2 ARDA

3.2.1 Dati Python

3.2.2 Dati Java

3.3 Valutazioni complessive

Come già detto nell'introduzione e nel capitolo 2, le varie modifiche apportate al codice hanno portato differenze marginali al livello di risultati complessivi. Il limitato miglioramento è dovuto alla dimensione delle aree prese in esame, troppo piccole per poter vedere i miglioramenti ottenibili delle nuove funzioni introdotte.

Questo però non va incriminato alla scelta dei dati ma proprio alla natura dei dati da analizzare, che cerca di prevedere le distanze su una serie di spostamenti ottenuti monitorando degli utenti nelle loro spostamenti abituali. Ed è proprio la natura ripetitiva dei spostamenti a rendere piccola la zona di interesse, visto che difficilmente una persona compie spostamenti giornalieri su un'area di centinaia di chilometri quadrati.

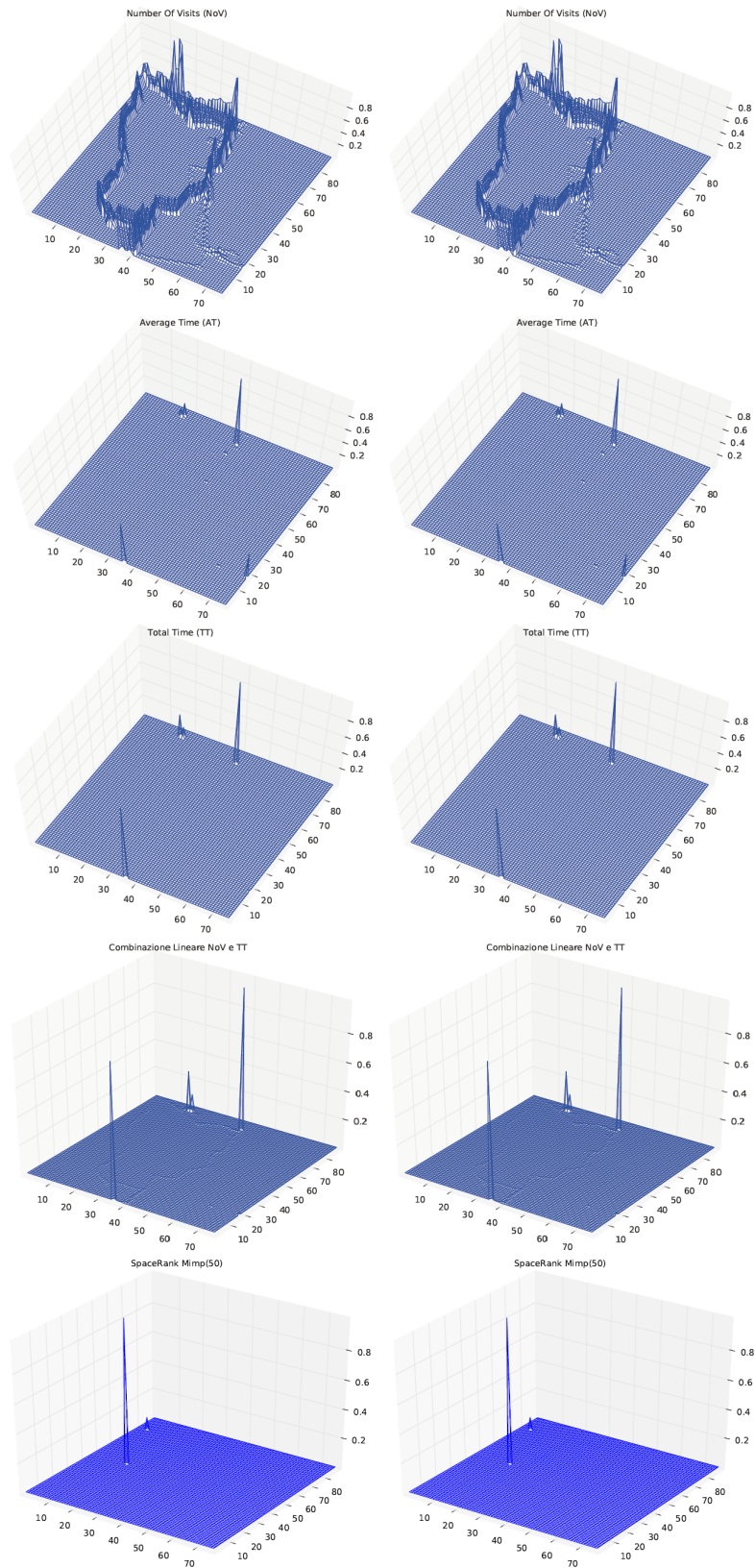


Figura 3.1: Esempio di attrazione gravitazionale

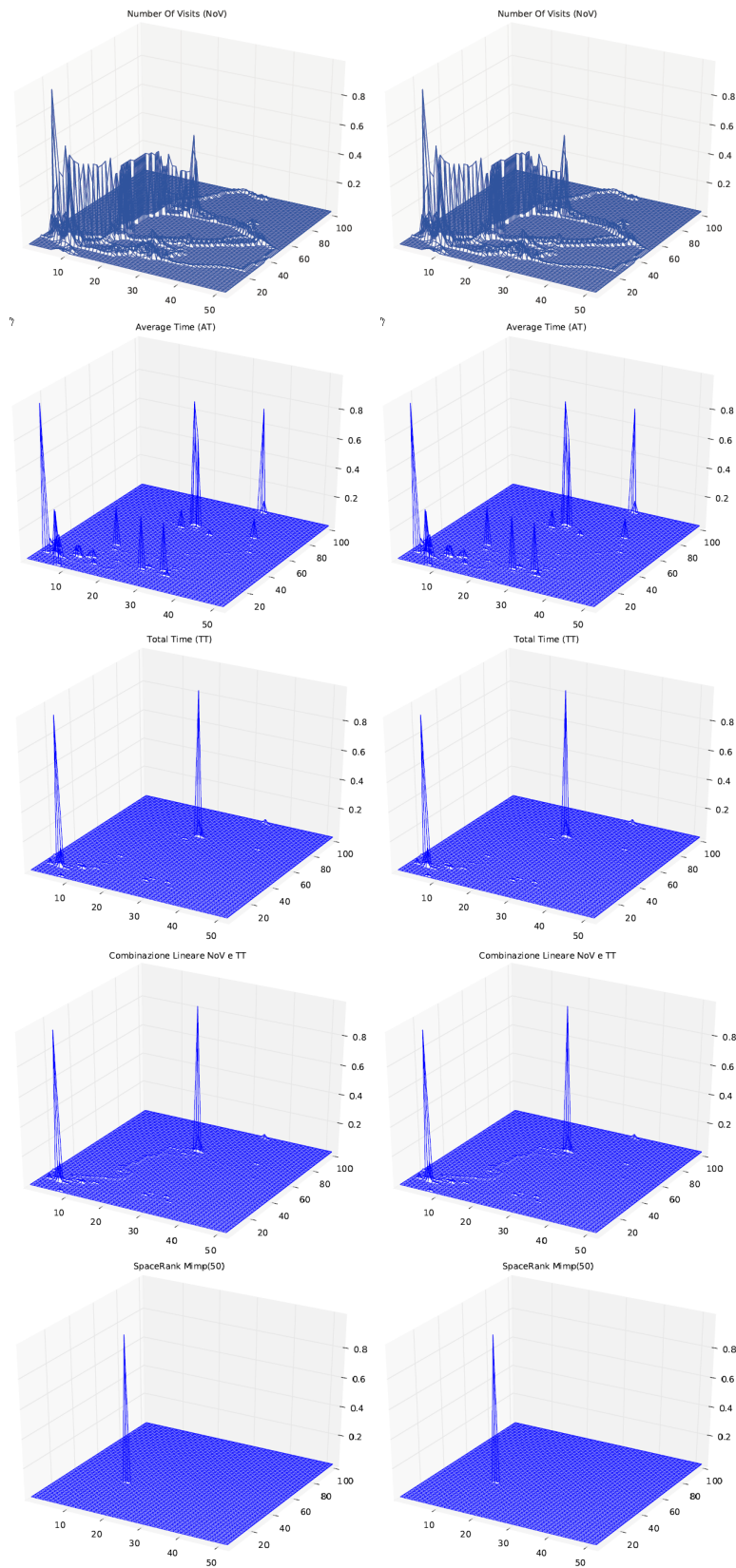


Figura 3.2: Esempio di attrazione gravitazionale

Capitolo 4

Comparative prestazionali

4.1 Confronto tra Java e Python

4.2 Confronto temporate

4.3 Test su dispositivi embedded

Capitolo 5

Conclusioni

In questa tesi si è voluto principalmente confrontare lo sviluppo dell'algoritmo ARDA con 2 linguaggi di programmazione concettualmente diversi tra loro: Java e Python. In seguito si è voluto apportare dei miglioramenti:

1. Si è voluto riorganizzare il codice per una maggiore fruibilità e leggibilità
2. Migliorare il calcolo della distanza tra punti e calcolo della matrice M_{acc}

Come ultima parte, si è voluto implementare un'interfaccia grafica per poter visualizzare i risultati in modo semplice e veloce, senza dimenticare la possibilità di esportare i dati in file CSV

5.1 Considerazioni

La conversione del codice da Python a Java, dopo l'attenta analisi del codice e di tutte le sue funzionalità e varianti, è stata abbastanza immediata, non trovando procedure o funzioni differenti tra i due linguaggi. Invece lo sviluppo dell'interfaccia grafica ha richiesto molta più attenzione. Java non dà nativamente le possibilità di sviluppare interfacce tramite dei tool esterni senza generare codice pesante e alle volte molto complesso a fronte della semplicità dell'interfaccia. Lo stesso sistema di gestione dei componenti e della loro dimensione e posizione non è intuitivo. Confrontandolo con linguaggi simili come C++ e lo stesso Python che, mediante librerie grafiche specifiche danno la possibilità di creare interfacce grafiche, Java ha una gestione abbastanza macchinosa per quando riguarda la composizione delle interfacce grafiche. Stesso discorso vale per la mancanza di vere e proprie librerie per la gestione dei grafici senza doversi appoggiare a libreria esterne a quelle di Java stessa. Queste mancanze non hanno comportato criticità nello sviluppo del programma ma non hanno nemmeno agevolato lo sviluppo della parte grafica.

5.2 Sviluppi futuri

Bibliografia

- [1] A.Dey, *Understanding and using context*, Personal and Ubiquitous Computing, vol. 5, no. 1, pp 4-7, 2001
- [2] Chaoming Song, Zehui Qu, Nicholas Blumm, Albert-Laszlo Barabasi. *Limits of Predictability in Human Mobility*, Science 19 February 2010: Vol.327 . no 5968, pp.1018-1021
- [3] Luca Snaidero, *Valutazione sperimentale dell'algoritmo ARDA per la previsione di traiettorie*, 2009
- [4] J. Krumm and E. Horvitz. *Predestination: Inferring destinations from partial trajectories ubicomp*. In Springer, editor, Lecture Notes in Computer Science, volume 4206, pages 243-260, 2006.
- [5] D. Ambrosin and A. Sciomachen. *A gravitational approach for locating new services in urban areas*. In Proceedings of the 16th Mini - EURO Conference and 10th Meeting of EWGT, pages 194-199, Poznan, Poland, 2005. Poznan University of Technology.
- [6] M. C. Gonzalez, C. A. Hidalgo, and A. L. Barabasi. *Understanding individual human mobility patterns*. Nature, 435(7196):779-782, 2008.
- [7] L. Page, S. Brin, R. Motwani, and T. Winograd. *The pagerank citation ranking: Bringing order to the web*. Technical report, Stanford Digital Library Technologies Project, 1998.
- [8] S. De Sabbata, *Pre-destinazione: modelli ed esperimenti per la previsione di traiettorie*, Tesi laurea specialistica Università degli Studi di Udine.
- [9] *The Java(tm) Language Environment*
<http://java.sun.com/docs/white/langenv/Simple.doc2.html>