Optimització i personalització del càlcul de rutes en dispositius mòbils

Ferran Ollé Pla
Universitat Oberta de Catalunya
Email: ferrano@uoc.edu

***Resum*— Actualment és fàcil trobar museus i zones turístiques que ens ofereixen un servei mitjançant el qual podem fer servir el nostre telèfon mòbil com a eina per guiar-nos durant la visita. Alguns utilitzen sistemes de geoposicionament per mostrar-nos la informació adient al punt on ens trobem.**

Tanmateix, aquests sistemes no tenen en compte les necessitats ni peculiaritats pròpies de cada usuari [1]. Algunes estan basades en que l'usuari té connexió a Internet, però s'hauria de tenir en compte que hi ha zones o llocs on la cobertura GSM és inexistent, per tant els mapes i informació hauria d'estar integrada en l'aplicació. D'altres tampoc tenen en compte els horaris de les instal·lacions a l'hora de crear la ruta per l'usuari, així doncs pot ser que el sistema ens faci anar a un museu que està tancat perquè és hora de dinar en lloc de guiar-nos cap a un restaurant. Cal, per tant, no tan sols oferir un servei de guia per dispositius mòbils a l'usuari, sinó que aquest servei s'hi adapti.

Aquest estudi es centra en trobar la ruta que millor satisfaci a les necessitats i preferències de l'usuari (personalització). Així, partint d'algorismes de routing existents, caldrà estudiar quin és el millor algorisme per implementar la funció de cost que doni la "bellesa" de la ruta per l'usuari.

***Paraules clau*— android, openstreetmap, osmdroid, personalization, routing.**

I. Introducció

En l'actualitat l'ús de telèfons mòbils de tipus smartphone està molt estès. Les grans companyies (Apple, HTC, Samsung, BlackBerry, Nokia, ...) treuen nous models cada pocs mesos i aquests no tenen res a veure amb els dispositius que teníem fa uns anys quan va aparèixer la telefonia mòbil a Espanya. Des de llavors el nombre d'usuaris d'aquesta tecnologia ha anat augmentant fins al punt de superar en nombre d'habitants el 31 de març de l'any 2006.

Aquests smartphones tenen capacitats multimèdia gràcies a les grans pantalles multi tàctils, càmeres d'última generació, acceleròmetres, sensors de llum i de posició, connexió a Internet d'alta velocitat i també receptor GPS. Rere el negoci dels smartphones hi ha el de les descàrregues d'aplicacions (gratuïtes i de pagament). Existeixen aplicacions capaces de fer qualsevol tasca inimaginable fa uns anys en un dispositiu mòbil. Per exemple podem trobar aplicacions que llegeixin codis de barres mitjançant la càmera de fotos, identifiquin el títol i autor d'una cançó només escoltar-la o permetin indicar a l'usuari quins restaurants hi ha a prop de l'actual posició utilitzant els satèl·lits GPS.

En aquest estudi ens centrarem en les aplicacions que utilitzen el receptor GPS del dispositiu per oferir serveis a l'usuari. Per exemple existeix l'aplicació Layar [2] que utilitza el receptor GPS i la càmera per oferir una experiència de realitat augmentada a l'usuari, permet afegir capes a l'aplicació per ubicar restaurants, atraccions turístiques, fotografies geolocalitzades, Tweets geolocalitzats. Molt semblant a aquesta però més pràctica existeix l'aplicació Google Maps, pre-instal·lada en tots els dispositius mòbils amb Android [3], que ens permet, a part de trobar tota mena de punts d'interès a prop d'on estem ubicats, calcular la ruta més apropiada entre dos punts segons les preferències de l'usuari (tipus de transport, tipus de carretera, ...).

El problema que existeix en totes aquestes aplicacions és la dependència de la connexió a Internet. Existeixen aplicacions per dispositius mòbils basades en els Sistemes d'Informació Geogràfica [4], com per exemple gvSIG Mini [5]. Aquesta aplicació és una adaptació de l'aplicació gvSIG Desktop per a dispositius mòbils amb sistema operatiu Android i també per a terminals compatibles amb Java. Permet la navegació online i offline de mapes de tot el món, així com la cerca de localitats, càlcul de rutes de punt a punt. Per realitzar la navegació offline s'utilitzen els mapes OpenStreetMap (OSM) [6], que es tracta d'un projecte col·laboratiu per crear mapes lliures de tot el món. També dins de (OSM) existeix una base de dades de punts d'interès turístic que ens serà útil per obtenir dades reals del territori. Existeixen també biblioteques específiques, com per exemple osm-droid [7] per integrar en aplicacions mòbils Android i que utilitzen l'estàndard obert d'OpenStreetMap, aquesta ens serà útil per confeccionar la nostra aplicació.

Una altra problemàtica que existeix en aquest tipus d'aplicacions és la personalització. Sovint el programa es comporta de la mateixa manera independentment dels gustos de l'usuari o restriccions horàries. Per exemple si a l'usuari en qüestió no li agraden els museus, la ruta que ens hauria d'oferir el sistema no hauria d'incloure els museus, o si és celíac només hauria de recomanar restaurants aptes per a celíacs. Aquest problema s'ha estudiat a fons en [1] i [11], presentant molt detalladament la problemàtica i presentant una possible solució utilitzant ontologies per guardar i gestionar les preferències. Altres estudis relacionats amb la personalització de rutes hem trobat [8] que es centra en analitzar les diferents opcions que existeixen per implementar aplicacions geogràfiques semàntiques en dispositius mòbils exposant per cada una d'elles els avantatges i inconvenients, i recomanant la més adient per cada cas. En [10] també trobem un estudi enfocat en la personalització de rutes per diferents tipus de usuaris i finalment ens presenta un prototip de la plataforma web i també una versió mòbil. Per tractar la gran quantitat

d'informació que necessitem per personalitzar les rutes necessitem ontologies perfectament definides i a [9] ens presenta un conjunt d'ontologies dedicades a guardar informació turística.

Utilitzant dades cartogràfiques és immediat trobar la distància entre dos punts (dues ciutats), però quan volem trobar la distància més curta en un seguit de punts (ciutats, llocs d'interès, etc.) tenim un problema. Aquest problema es coneix com a Travelling Salesman Problem (TSP), un subconjunt del problema anomenat Vehicle Routing Problem (VRP). Aquest tipus de problemes pertanyen a la classe NP-Complet de la teoria de la complexitat computacional i els trobem molt ben descrits al document tècnic [12]. En aquest document ens presentarà la problemàtica existent amb els problemes TSP, VRP i totes les seves variants així com les maneres de resoldreho utilitzant diferents algorismes heurístics i meta-heurístics.

Un dels algorismes més utilitzats per resoldre problemes de cerca de rutes són els Algorismes Genètics [13]. Existeixen vàries biblioteques especialitzades per desenvolupar aplicacions utilitzant GA's com per exemple [14]. També a [15] es proposa un algorisme genètic per resoldre la navegació entre dos punts d'un mapa utilitzant un dispositiu mòbil tenint en compte dades de transit i límits de velocitat de les carreteres. A [16] ens proposen un sistema personal de navegació, per guiar turistes a través de múltiples destinacions eficientment. Aquest té en compte restriccions de temps dels usuaris i el seu grau d'interès. Aquí també es proposa la utilització d'un algorisme genètic, però en aquest cas, el seu càlcul l'efectuarà un servidor extern i el dispositiu mòbil el rebrà a través d'Internet utilitzant un web service.

També existeixen altres tipus d'algorismes per realitzar rutes, com per exemple els basats en colònies de formigues tal i com veiem a [17]. En aquest article podrem veure un possible algorisme per utilitzar en la nostra aplicació.

Els problemes de tipus NP-Complet es caracteritzen per necessitar elevats requeriments de maquinari (velocitat de processador i memòria) i és per això que ejecutar-los en dispositius mòbils on tenim capcitat de càlcul limitada és tot un repte.

Amb tot el que hem vist ens podem fer una idea del que necessitarem per desenvolupar la nostra aplicació que calculi rutes per diferents punts d'interès tenint en compte les preferències dels usuaris i utilitzant un dispositiu mòbil.

El document s'estructura de la següent manera:

- A la secció 2 es proven diferents algorismes per elaborar rutes, ja sigui utilitzant alguna de les biblioteques trobades o implementant nosaltres mateixos el codi necessari.
- Seguidament a la secció 3 s'analitzen les dades que es necessiten recollir dels usuaris per tal de poder personalitzar les rutes amb els seus gustos, preferències i disponibilitats horàries.
- A la secció 4 s'avaluen el mínim de dades que es necessiten guardar en el telèfon mòbil per poder oferir un servei correcte a l'usuari.
- A la secció 5 es presenta l'aplicació que s'ha creat (figura 1) per realitzar rutes turístiques personalitzades als usuaris, sense la necessitat de connexió a Internet.

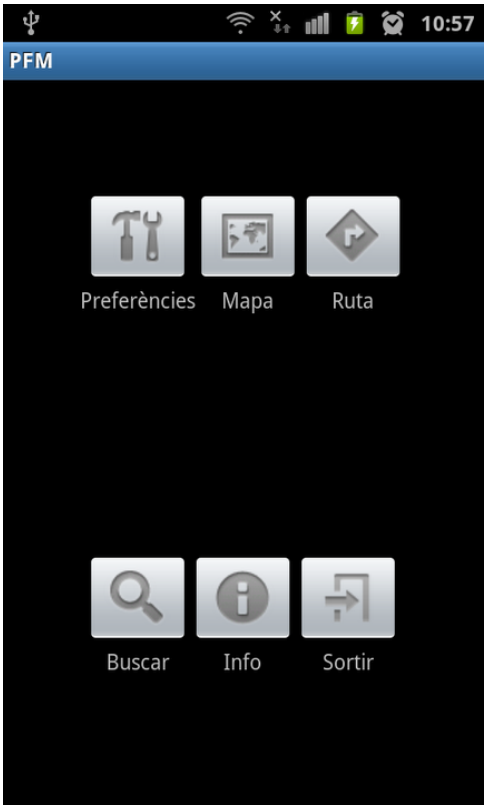

Fig. 1. Aplicació Android desenvolupada.

- I finalment a la secció 6 s'exposen les conclusions de la investigació realitzada així com possibles treballs futurs.

II. Algorismes de Routing

Tal com hem vist a [12] existeixen diferents algorismes per resoldre el problema del viatjant de comerç (TSP) o el problema de les rutes de vehicles (VRP), que al cap i a la fi són el mateix problema: calcular el mínim temps necessari, o distancia mínima per visitar diferents punts d'interès turístic en una ciutat.

Podem classificar els algorismes en:

- Exactes/Òptims
- Algorismes Heurístics
- Algorismes Meta-heurístics

A. *Exactes/Òptims*

Els algorismes exactes o òptims són aquells que busquen en l'espai de cerca totes les possibles solucions al problema que s'està resolent i acaben donant la millor solució que existeix. Això ens assegura que sempre trobarem la solució òptima, però el cost d'aquests algorismes sol ser massa alt sobretot si estem pensant implementar-los en un dispositiu mòbil amb poca capacitat de procés i memòria.

B. *Algorismes Heurístics*

Les tècniques heurístiques són algorismes que troben solucions de bona qualitat en problemes combinatoris complexes

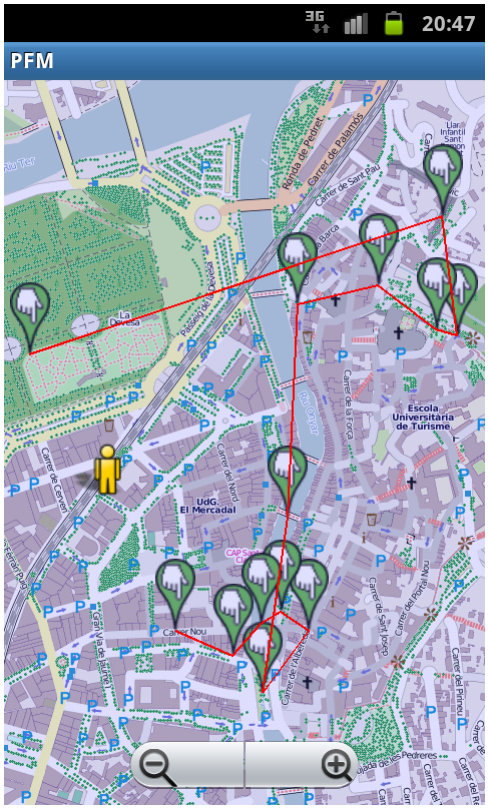

Fig. 2. Ruta per tots els POI utilitzant Nearest Neighbor

explorant el coneixement del domini, es a dir, estan especialment dissenyats per resoldre un problema concret. Normalment són fàcils d'implementar i troben bones solucions amb esforços computacionals relativament petits, però per altra banda, renuncien a trobar la solució òptima global del problema. En problemes grans rarament un algorisme heurístic trobarà una bona solució.

El el nostre cas pot ser interessant la utilització d'aquests tipus d'algorismes, ja que ens oferiran una solució ràpida al problema que tenim sense grans requeriments de procés de càlcul i memòria.

L'algorisme heurístic més conegut és el *Nearest Neighbor* (Veí més proper) [18].

1) Nearest Neighbor Algorithm: L'algorisme del veí més proper és un dels més simples i fàcils de construir entre els algorismes heurístics. La solució es construeix de la següent manera:

- S'escull un punt al atzar de l'espai de cerca. (en el nostre cas serà la posició actual obtinguda mitjançant el GPS)
- Després s'escull el punt més pròxim al actual que encara no s'hagi visitat i així successivament fins que s'hagin visitat tots els punts.

A la figura 2 podem veure com quedaria una possible ruta per tots els punts d'interès utilitzant l'algorisme del veí més proper. A simple vista es pot veure que la ruta és prou bona encara que no és la solució òptima al problema.

Per el nostre problema en concret, a part de minimitzar les distàncies i temps entre punts, també hem de complir unes preferències horàries (hora d'inici i fi de la ruta), restriccions temporals (horaris dels museus o els punts d'interès) i preferències culinàries (tipus de menjar i hora que vol dinar) dels usuaris de l'aplicació.

Per tant l'algorisme quedaria de la següent manera:

$NodeActual \leftarrow PosicioActualGPS$;
$AfegirPuntRuta(NodeActual)$;
while *quedin punts per visitar* **do**
  if *VolDinar i no(HemDinat) i EsHoraDeDinar* **then**
    $Restaurant \leftarrow$
    $BuscarRestaurant(NodeActual, PrefCulinaries)$;
    $NodeActual \leftarrow Restaurant$;
    $AfegirPuntRuta(Restaurant)$;
    $HemDinat \leftarrow true$;
  end
  $SeguentNode \leftarrow$
  $BuscarNodeMesProximNoVisitat(NodeActual)$;
  Calcular Temps i Distàncies recorregudes;
  if *el temps de ruta no excedeix del limit* **then**
    if *esta obert ?* **then**
      Actualitzar Temps i Distàncies recorregudes;
      $MarcarPuntComAVisitat(SeguentNode)$;
      $AfegirPuntRuta(SeguentNode)$;
      $NodeActual \leftarrow SeguentNode$;
    end
  else
    Mostrar Missatge que no es podran visitar tots els punts;
  end
end

Algorithm 1: Nearest Neighbor Algorithm

C. Algorismes Meta-Heurístics

Els algorismes meta-heurístics són algorismes de propòsit general, que no depenen del problema, i donen bons resultats però no acaben oferint la solució òptima sinó solucions subòptimes. S'acostumen a utilitzar per aquells problemes que no existeix un algorisme o heurística específica que els resolgui.

Existeixen moltes implementacions i híbrids, els més importants són:

- GA: Genetic Algorithms (Algorismes Genètics)
- TS: Tabu Search (Cerca Tabú)
- SA: Simulated Annealing
- ACO: Ant Colony Optimization (Optimitzacions amb colònies de formigues)

Per resoldre un problema semblant al nostre s'han utilitzat algorismes Genètics en els treballs [15] i [16], en aquests treballs es donava suport de càlcul en el dispositiu mòbil mitjançant un servidor extern (a través d'Internet) ja que aquests algorismes requereixen gran capacitat de càlcul per trobar una bona solució en poc temps. A nosaltres ens interessa

poder executar aquests tipus d'algorismes en el dispositiu mòbil, per tant pot ser que no sigui viable utilitzar-los.

S'han realitzat proves amb la biblioteca JAGA [14], executant el mateix test en un equip d'escriptori i un dispositiu mòbil.

Les característiques de l'equip d'escriptori són:

- Processador Core 2 Duo E6850 , 3.00 GHz
- 3GB de memòria RAM
- S.O. Windows 7 Professional

Les del dispositiu mòbil:

- Samsung Galaxy S i9000
- Processador ARM Cortex A8 Hummingbird 1GHz, GPU PowerVR SGX540
- 512 MB RAM, 2GB ROM
- OS Android 2.3.6 Gingerbread

Aquestes proves no han estat gaire esperançadores ja que el temps d'execució del mateix algorisme en una i altra plataforma han estat molt diferents:

- Equip d'escriptori: 3 segons
- Dispositiu mòbil: 2 min 33 segons.

Tot i ser un dispositiu mòbil d'última generació amb gran capacitat de càlcul, la poca memòria RAM que disposa fa que aquests tipus d'algorismes no es puguin executar en un temps considerable per l'usuari.

III. Personalitzacio

Com hem vist en la secció II la personalització intervé en una part important de l'algorisme de routing.

Concretament en la nostra aplicació (figura 3) tindrem en compte els següents punts:

- Hora d'inici i fi de ruta
- Tipus de punts d'interès que es volen visitar
- L'usuari vol dinar o no
- Tipus de restaurant
- Hora que l'usuari vol parar-se per dinar

Per calcular els temps de ruta també intervenen altres factors:

- Distància i temps per anar d'un punt a un altre
- Temps necessari per realitzar la visita al punt.

Finalment també es tindran en compte els horaris d'obertura dels punts d'interès, per saber si val la pena dirigir a l'usuari en un punt concret que estarà tancat.

És per aquest motiu que al dissenyar la base de dades on s'emmagatzemarà tota la informació relativa als punts d'interès també s'han afegit camps especials per poder tenir tot aquest coneixement. A la taula I podem veure com quedaria l'estructura i els camps que la composen. Els punts d'interès estan classificats segons un tipus tal i com veiem a la taula II. I els de tipus restaurant estan també sub-classificats en tipus de restaurant com veiem a la taula III.

El primer filtre que tindrem són els tipus de punts d'interès que l'usuari voldrà visitar, segons les seves preferències l'espai de cerca es veurà reduït o ampliat.

El segon filtre dependrà de les preferències horàries de l'inici i el final de la ruta, d'aquesta manera sabrem les hores (o minuts) totals disponibles per realitzar la ruta. L'hora inicial

Fig. 3. Pantalla de preferències de l'aplicació

poi table		
id	INTEGER	Identificador
type	NUMERIC	Tipus de punt
lat	NUMERIC	Latitud
lon	NUMERIC	Longitud
name	TEXT	Nom
address	TEXT	Adreça
phone	PHONE	Telèfon
timetable	TEXT	Horari
visittime	NUMERIC	Temps estimat de visita
rest_type	NUMERIC	Tipus de restaurant

TABLE I

Taula de punts d'interès

id	type
1	Place of Worship
2	Museum
3	Historical
4	Park/Garden
5	Restaurants
6	Pharmacy

TABLE II

Tipus de punts d'interès

id	rest_type
1	REGIONAL
2	ITALIAN
3	FRENCH
4	CHINESE
5	JAPANESE
6	TAPAS
7	FAST FOOD

TABLE III

Tipus de menjar

i final de ruta està composta per minuts i hores, per tant per poder realitzar els càlculs entre hores treballarem tots els valors en minuts.

$$MinutsInicial = 60 * HoraInicial + MinutsInicial \quad (1)$$
$$MinutsFinal = 60 * HoraFinal + MinutsFinal \quad (2)$$
$$TempsTotalRuta = MinutsFinal - MinutsInicial \quad (3)$$

D'aquesta manera a TempsTotalRuta tindrem tots els minuts disponibles per crear la ruta personalitzada a l'usuari.

A partir d'aquí, i utilitzant l'algorisme del veí més proper vist en la secció II, es crearà la ruta personalitzada amb les preferències de l'usuari.

IV. Anàlisis i tractament de dades

Un dels objectius d'aquest treball és la d'analitzar el mínim de dades necessàries que necessitem tenir en el telèfon mòbil per poder oferir un servei correcte a l'usuari i amb un temps de resposta raonablement curt. Aquestes dades han de ser accessibles sense connexió a Internet, per tant s'hauran d'emmagatzemar a la memòria interna del telèfon.

Necessitem tenir a l'abast:

- Cartografia de mapes
- Punts d'interès

Tota aquesta informació la tenim gràcies a OpenStreetMap [6]. El que haurem de fer es tractar-la de manera eficient perquè es pugui treballar sense connexió a Internet des del telèfon mòbil.

Primer de tot, però, s'han analitzat diferents biblioteques preparades per dispositius Android [3]:

- OsmAnd [19] - Map Viewing and Navigation based on Open Street Maps for mobile devices.
- OSMDroid [7] - OpenStreetMap Tools for Android.

A. Projectes basats en OSM per dispositius mòbils Android

1) OsmAnd: Es tracta d'una aplicació creada per Victor Schreb. Està llicenciada com a GPL i existeix una versió gratuïta limitada i una versió de pagament amb més funcionalitats. De totes maneres tambe permet la descarrega del seu codi font per poder utilitzar en la nostra aplicació.

Avantages:

- Visualització de mapes online i offline (vectorials i amb tesel·les)
- Navegació online i offline per carrers (routing).
- Punts d'interès provinents de OpenStreetMap online i offline.
- Permet la descarrega de mapes sencers (carrers, noms, punts d'interes, ...)

Inconvenients:

- Dificultat d'integrar en les nostres aplicacions.
- La navegació offline per carrers no esta prou depurada.

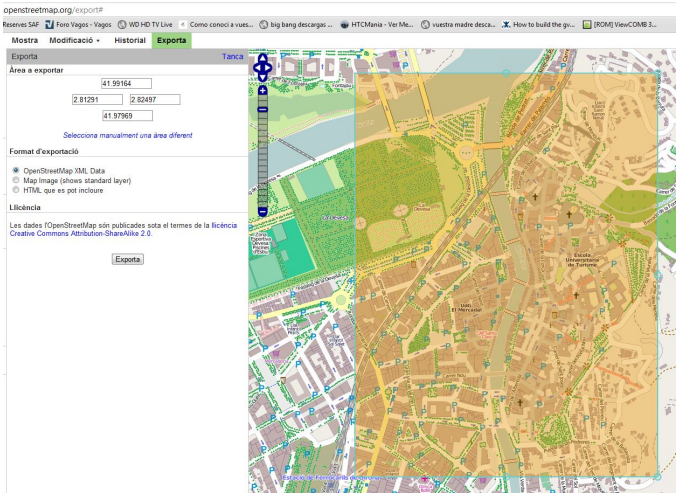

Fig. 4. Zona capturada mitjançant la web d'OSM

2) OSMDroid: Es tracta d'una aplicació creada per Nicolas Gramlich per la gestió i visualització d'informació geogràfica en dispositius mòbils Android. Està llicenciada com a LGPL i segons anuncien en la seva web, reemplaça les funcionalitats de la classe MapView nativa d'Android, és a dir, pot realitzar pràcticament les mateixes funcions però amb tots els avantatges d'utilitzar OpenStreetMap i no Google Maps [20] com a base geogràfica.

S'ha utilitzat en gran quantitat d'aplicacions existents gràcies a la fàcil integració i programació que ofereix als desenvolupadors. Està encapsulada en una biblioteca .jar i existeixen bastants tutorials i exemples per Internet.

Avantatges:

- Fàcil integració gràcies a la biblioteca proporcionada.
- Visualització de mapes online i offline (mitjançant tesel·les)

Inconvenients:

- No permet la navegació per carrers offline (routing)

Una de les grans diferències que trobem entre OsmAnd i OSMDroid és que un pot utilitzar mapes en format vectorial permetent obtenir rutes entre punts de mode offline, però la dificultat d'integrar la nostra aplicació amb OsmAnd i al no estar depurat el sistema de routing offline, ens decantem per OSMDroid. Ens oferirà estabilitat, fiabilitat i molts recursos útils per realitzar la nostra aplicació.

B. Obtenció de dades d'OpenStreetMap

Existeixen vàries maneres d'obtenir informació de OpenStreetMap [6], recursos online, descàrrega directa de mapes, biblioteques especialitzades,... Però la manera més fàcil, sobretot si volem treballar només una petita zona d'un mapa, és la utilització de la mateixa pàgina web d'OSM ja que té la funcionalitat d'exportar zones de mapes.

Nosaltres al treballar en una aplicació turística ens centrarem en una zona, concretament la ciutat de Girona. Com veiem a la figura 4 s'ha extret la informació de la zona cèntrica de la ciutat ja que allà és on hi han els punts d'interès turístic.

Per altra banda, l'eina d'extracció d'informació de la web d'OSM no permet extreure mapes de grans dimensions i el mapa de Girona ha estat creat amb gran detall i exactitud per tant no podem extreure tota la ciutat.

De totes aquestes dades obtingudes en format OSM-XML [21] s'extraurà informació relativa als punts d'interès i també als carrers de la ciutat per poder oferir a l'usuari detall de les rutes i distàncies a recórrer.

1) Punts d'interès: Per extreure dades relatives als punts d'interès de la zona, s'han seguits els passos indicats en la pàgina web de Alastair Aitchison [22].

Gràcies a les funcions espacials de SQL Server i als scripts proporcionats a la web hem pogut emplenar la taula de punts d'interes I, de totes maneres les dades que ens han feien servei de OSM-XML han estat:

- Nom
- Latitud
- Longitud

Això vol dir que la resta de dades necessàries les hem hagut d'entrar manualment com per exemple:

- Tipus de punt d'interès
- Adreça
- Telèfon
- Horari d'obertura
- Tipus de restaurant

També s'han creat nous punts d'interès ja que a OSM no hi eren tots. Per tant, com que treballarem en una aplicació turística de la ciutat de Girona , s'han afegit els llocs més populars i atractius per els visitants amb totes les dades necessàries.

La base de dades de punts d'interès comptarà amb un total de 104 punts diferents repartits en les diferents categories.

2) Carrers i carreteres de la ciutat: Per calcular la distància entre dos punts d'interès es pot fer d'una manera bàsica mitjançant la formula del Haversine [23] que està implementada en la biblioteca de OSMDroid [7]. Aquesta ens retornara la distancia en metres existent entre dos punts en línia recta tenint en compte la curvatura de la terra.

Està clar que la distancia no serà real, ja que a no ser que siguin dos punts en un descampat, existiran tota mena d'obstacles per arribar d'un punt a l'altre. Així que per fer-ho exacte haurem de tenir en compte els carrers de la ciutat per calcular aquesta distància.

Una ciutat es pot representar com un graf geomètric on les interseccions dels carrers formen els vèrtexs i els carrers que uneixen les interseccions són les arestes. Per trobar la distància mínima existent entre dos punts de la ciutat o vèrtexs del graf s'utilitza normalment l'algorisme de Dijkstra [24].

Aquest algorisme està implementat en moltes biblioteques especialitzades per el càlcul de rutes entre punts, entre elles:

- Traveling Salesman [25]: Es tracta d'una llibreria en Java que també incorpora una aplicació visual per carregar dades d'OSM i realitzar proves de càlculs de rutes entre punts. Per altra banda també ofereix una solució al problema del viatjant de comerç.

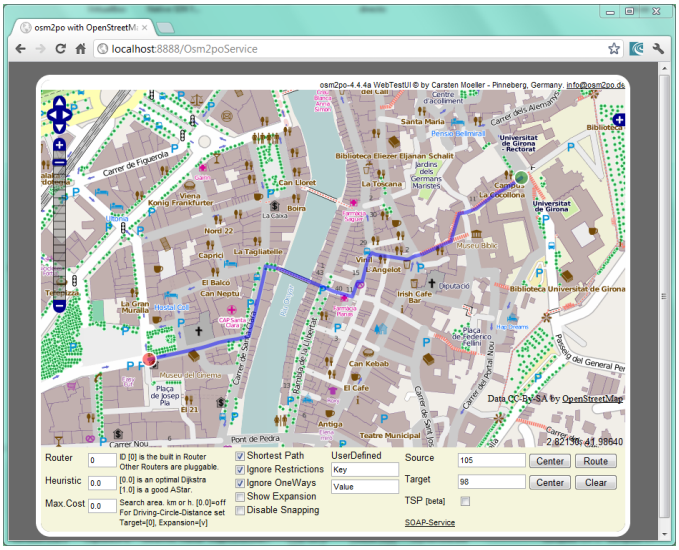

Fig. 5. OSM2PO routing WebTestUI

- OSM2PO [26]: Desenvolupat en Java. Converteix dades d'OSM-XML en grafs dirigits per poder aplicar els algorismes de routing. Capaç de convertir mapes grans. Permet generar fitxers SQL compatibles amb PostGIS. Incorpora un servidor web per realitzar proves de rutes molt atractiu visualment com veiem a la figura 5.

Cal dir que les dues biblioteques no estan optimitzades per ser executades en la plataforma Android, però s'ha provat d'incorporar-les a la nostra aplicació amb diferents resultats.

Amb la biblioteca Travelling Salesman no s'ha arribat a executar l'aplicació, segurament perquè les dependències són massa grans i la màquina virtual de Android no accepta una biblioteca d'aquestes dimensions.

Amb OSM2PO tot el contrari, no només hem aconseguit l'execució de la biblioteca correctament, si no que ofereix resultats realment bons i sobretot amb uns temps de resposta molt i molt baixos (uns mili-segons). La integració amb l'aplicació Android és realment fàcil gràcies als exemples que ofereix la biblioteca.

V. Prototip Android

S'ha dissenyat una aplicació per un dispositiu mòbil Android [3] per poder realitzar proves reals del funcionament dels algorismes i la personalització als usuaris.

L'equip de proves té les següents característiques:

- Samsung Galaxy S i9000
- Processador ARM Cortex A8 Hummingbird 1GHz, GPU PowerVR SGX540
- 512 MB RAM, 2GB ROM
- Pantalla SuperAMOLED multi-tàctil, 480 x 800 píxels de 4"
- GPS amb suport A-GPS
- Brújula digital
- 3G HSDPA / HSUPA
- Wi-Fi 802.11 b/g/n
- OS Android 2.3.6 Gingerbread

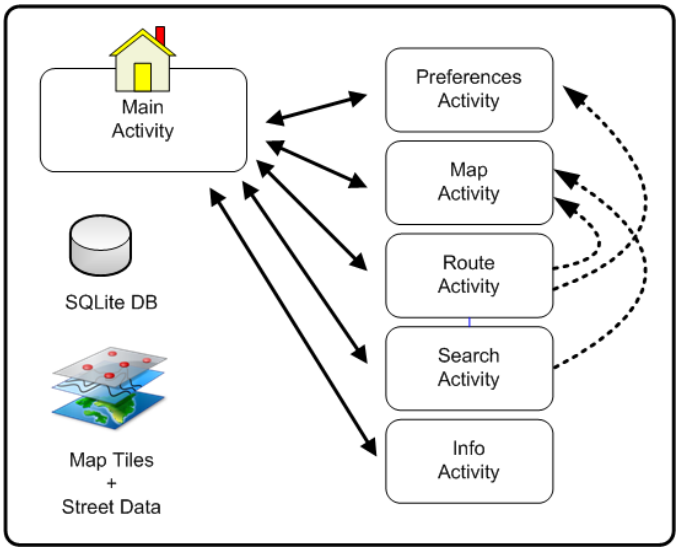

Fig. 6. Mapa de l'aplicació.

Per el desenvolupament de l'aplicació s'han utilitzat les següents biblioteques de tercers:

- OSMDroid 3.0.8 [7]
- OSM2PO 4.4.4a [26]
- SLF4J Android 1.5.8 [27] (necessària per OSMDroid)

L'aplicació està estructurada en les següents activitats (figura 6):

A. Activitat: Menú principal

Activitat des d'on s'accedirà a la resta d'activitats de l'aplicació (figura 1). En aquesta activitat també es crearan les connexions necessàries a la base de dades SQLite.

B. Activitat: Preferències

Tota la gestió de les preferències de l'usuari (figura 3) estarà gestionada per la classe *PreferenceActivity* d'Android, per tant d'una manera ràpida i senzilla podrem accedir-hi des de qualsevol altra Activitat de l'aplicació.

C. Activitat: Mapa

Aquesta activitat permetrà visualitzar el mapa de la zona (figura 7) utilitzant la biblioteca OSMDroid [7]. Es mostrarà la ruta a seguir si s'ha creat prèviament i continuarà en el mateix punt d'interès on l'usuari havia acabat en l'anterior consulta al mapa. Mitjançant uns controls afegits en la part inferior de l'activitat, l'usuari podrà navegar per els diferents punts d'interès de la ruta. També ens permetrà visualitzar la ruta en format text mitjançant una llista desplegable. A més, prement qualsevol punt d'interès del mapa i gràcies a la biblioteca OSM2PO [26], podrem visualitzar la ruta més curta per arribar-hi.

D. Activitat: Ruta

En aquesta activitat (figura 8) podrem calcular una nova ruta, eliminar una ruta ja creada, visualitzar la ruta en el

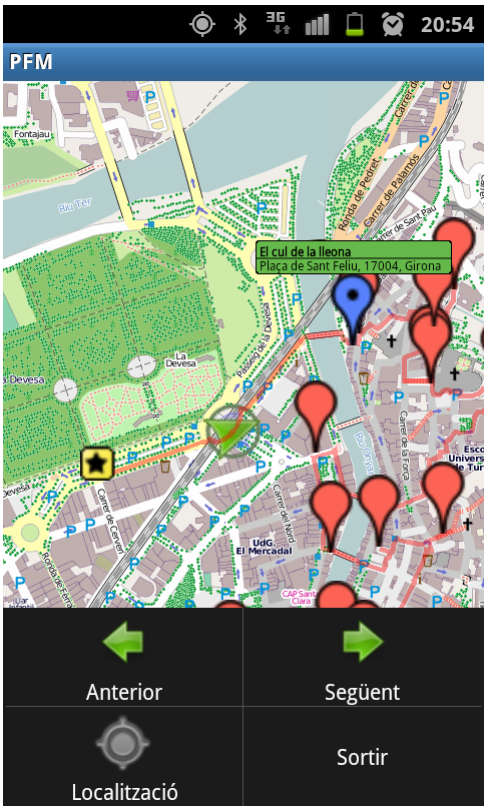

Fig. 7. Mapa de la ruta calculada.

Fig. 8. Activitat per gestionar les noves rutes.

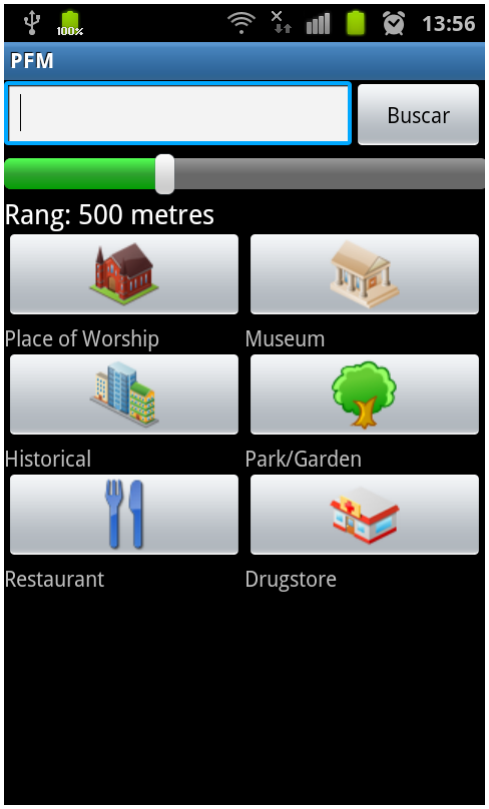

Fig. 9. Activitat de cerca de punts d'interès.

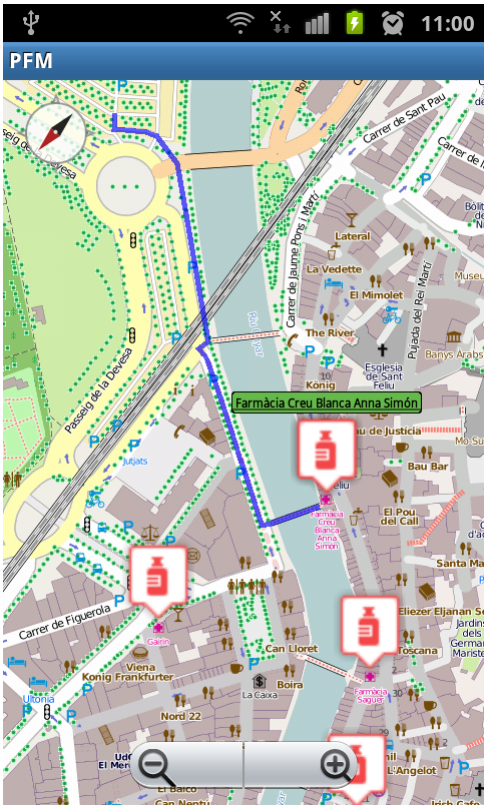

Fig. 10. Resultats obtinguts cercant farmàcies a 500m de l'usuari.

mapa o accedir al menú de preferències. S'accedirà al fitxer de preferències per saber les prioritats que te l'usuari per calcular la ruta, a la base de dades de punts d'interès per obtenir informació i a la biblioteca OSM2PO [26] per realitzar el càlcul de distancies exactes de tota la ruta.

E. Activitat: Buscar

Permet cercar (figura 9) diferents tipus de punts d'interès per categories o per nom. Permet fer una cerca radial tot partint de la ubicació de l'usuari. El resultat de la cerca es mostra sobre el mapa i l'usuari pot saber com arribar-hi prement a la icona corresponent tal i com es veu a la figura 10.

F. Comportament de l'aplicació en diferents escenaris.

S'han efectuat diverses proves amb l'aplicació per comprovar el seu correcte funcionament amb diferents preferències d'usuari:

1) Escenari 1:

- Hora inici: 11:00.
- Hora fi: 20:00.
- Tipus de POI que es vol visitar:
  - *Place of Worship*.
  - *Park/Garden*.
- Es vol parar a dinar: Si.
- Tipus de restaurant: *Japanesse*.
- Hora de dinar: 13:00.

Resultats:

- Parc de la Devesa (525m - 6 min).
- Església de Sant Feliu (381m - 5 min).
- Monestir de Sant Pere Galligants (260m - 3 min).
- Jardins de John Lennon (114m - 1 min).
- La Riba - Sushi Bar (560m - 7 min).
- Catedral de Girona (350m - 4 min).
- Claustre de la Catedral (15m - 0 min).
- Jardins de la Francesa (197m - 2min).

Total: 2402 metres.
Temps de ruta: 281 minuts (comptant el temps de desplaçament i de visita).

2) Escenari 2: Idèntics al Escenari 1 però l'usuari no vol aturar-se a dinar.
Resultats:

- Parc de la Devesa (525m - 6 min)
- Església de Sant Feliu (381m - 5 min)
- Monestir de Sant Pere Galligants (260m - 3 min)
- Jardins de John Lennon (114m - 1 min)
- Catedral de Girona (387 - 5 min)
- Claustre de la Catedral (15m - 0 min)
- Jardins de la Francesa (197m - 2min)

Total: 1879 metres.
Temps de ruta: 185 minuts.

A la figura 11 es poden veure les dues rutes, la primera amb la visita al restaurant japonès "La Riba - Sushi Bar" i la segona no.

3) Escenari 3:

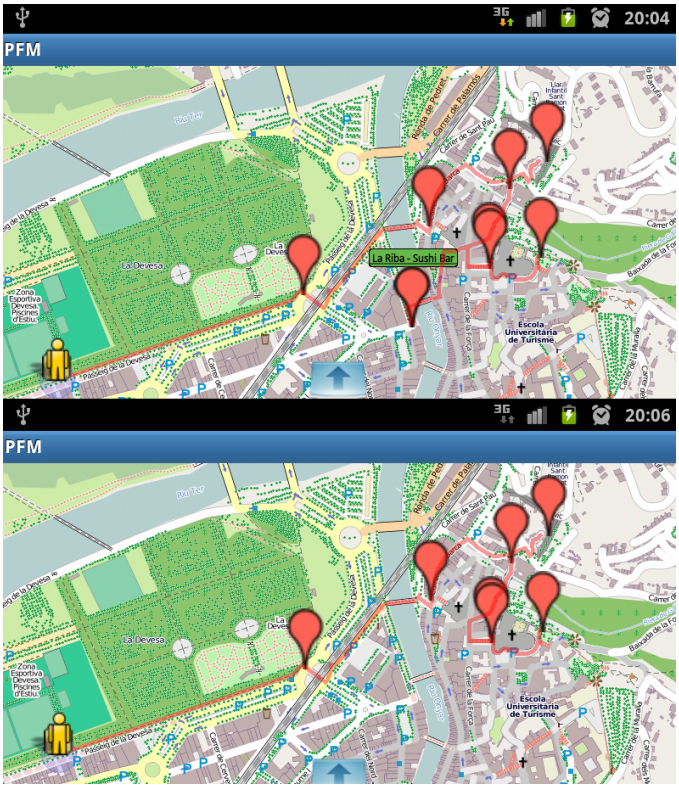

Fig. 11. Ruta obtinguda per diferents preferències d'usuari.

Fig. 12. Missatge que mostra l'aplicació al no disposar el temps suficient per completar una ruta.

- Hora inici: 13:00
- Hora fi: 16:00
- Tipus de POI que es vol visitar:
  - *Place of Worship*
  - *Historical*
- Es vol parar a dinar: Si
- Tipus de restaurant: *Any* (Qualsevol)
- Hora de dinar: 13:30

Resultats:

- Carrer Nou (568 - 7 min)
- Carrer de Santa Clara (107m - 1 min)
- Plaça de Catalunya (63m - 1 min)
- L'Albareda (113m - 1 min)
- Plaça del Ví (140m - 2 min)
- Cases de l'Onyar (50m - 1 min)
- Pont Effiel (265m - 3 min)
- Plaça dels Raïms (153m - 2 min)
- Palau de l'Agullana (154m - 2 min)
- Catedral de Girona (351m - 4 min)

Total: 1964 metres.
Temps de ruta: 173 minuts (comptant el temps de desplaçament i de visita).

En aquest exemple (figura 12) no es poden visitar tots els punts d'interès amb el temps que disposa l'usuari per fer la ruta, ja que només té tres hores per visitar la ciutat.

4) Escenari 4: Idèntics al Escenari 3 però l'usuari no vol aturar-se a dinar.

Resultats:

- Carrer Nou (568 - 7 min)
- Carrer de Santa Clara (107m - 1 min)
- Plaça de Catalunya (63m - 1 min)
- Cases de l'Onyar (96m - 1 min)
- Plaça del Ví (50m - 1 min)
- Plaça dels Raïms (266m - 3 min)
- Pont Effiel (153m - 2 min)
- Palau de l'Agullana (264m - 3 min)
- Catedral de Girona (351m - 4 min)
- Claustre de la Datedral (15m - 0 min)
- Banys Arabs (182m - 2 min)
- Monestir de Sant Pere de Galligants (106m - 1 min)
- Església de Sant Feliu (260m - 3 min)
- El Cul de la llegona (13m - 0 min)

Total: 2494 metres.
Temps de ruta: 175 minuts (comptant el temps de desplaçament i de visita).

A la figura 13 es pot veure com les rutes son completament diferents. Al no aturar-se a dinar es disposa de més temps per visitar la ciutat i per tant s'han afegit més punts d'interès a la ruta.

VI. Conclusions i treballs futurs

El resultat de la investigació és una aplicació Android completament funcional que permet calcular rutes turístiques de la ciutat de Girona sense la necessitat de tenir connexió a Internet, tenint en compte les preferències de l'usuari i els horaris d'obertura dels punts d'interès.

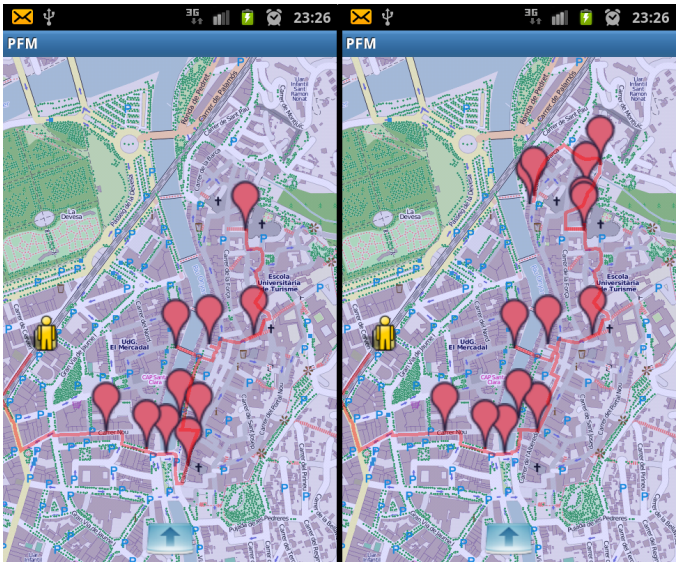

Fig. 13. Ruta obtinguda per diferents preferències d'usuari.

S'ha utilitzat l'algorisme heurístic del veí més proper [18] per realitzar les rutes, una base de dades SQLite per emmagatzemar informació relativa als punts d'interès turístic, la biblioteca OSMDroid [7] per visualitzar els mapes en el dispositiu mòbil i la biblioteca OSM2PO [26] per calcular les distancies i carrers per anar d'un punt d'interès a un altre.

Com a treballs futurs i per donar sortida comercial a l'aplicació, es podria complementar amb una pàgina web on els usuaris es poguessin descarregar nous mapes i informació turística de la zona que es vol visitar. També es podria afegir més informació de cada punt d'interès (fotos, descripció, peculiaritats,...) i permetre que els usuaris n'afegissin de nous.

REFERÈNCIES

[1] Descamps, L., Casas, J., Pérez, A., Conesa, J.,"Personalización de servicios basados en localización: un caso práctico." presented at the V Jornadas de SIG lliure, Girona (Spain) , March 24, 2011

[2] Layar site. [Last search: 16/05/2012]. Available: http://www.layar.com/

[3] Android SDK site. [Last search: 16/05/2012]. Available: http://developer.android.com/sdk/index.html

[4] Perez, Navarro Antoni, and Plana Albert Botella. "Introduccion a Los Sistemas De Informacion Geografica Y Geotelematica." Barcelona: Editorial UOC, 2011. Print.

[5] gvSIG Site. [Last search: 16/05/2012]. Available: http://www.gvsig.org/

[6] OpenStreetMap Site. [Last search: 16/05/2012]. Available: http://www.openstreetmap.org

[7] OpenStreetMap Tools for Android [Last search: 16/05/2012]. Available: http://code.google.com/p/osmdroid/

[8] Descamps, L., Casas, J., Pérez, A., Conesa, J.,"Cómo introducir semántica en las aplicaciones SIG móviles: expectativas, teoría y realidad." presented at the V Jornadas de SIG lliure, Girona (Spain) , March 24, 2011

[9] Descamps, L., Casas, J., Pérez, A., Conesa, J.,"Hacia la mejora de la creación de rutas turísticas a partir de información semántica" presented at the V Jornadas de SIG lliure, Girona (Spain) , March 24, 2011

[10] Ander Garcia, Olatz Arbelaitz, Maria Teresa Linaza, Pieter Vansteenwegen, and Wouter Souffriau. 2010. "Personalized tourist route generation." In Proceedings of the 10th international conference on Current trends in web engineering (ICWE'10), Florian Daniel and Federico Michele Facca (Eds.). Springer-Verlag, Berlin, Heidelberg, 486-497.

[11] Wouter Souffriau and Pieter Vansteenwegen. 2010. "Tourist trip planning functionalities: state-of-the-art and future." In Proceedings of the 10th international conference on Current trends in web engineering (ICWE'10), Florian Daniel and Federico Michele Facca (Eds.). Springer-Verlag, Berlin, Heidelberg, 474-485.

[12] José Antonio Rodríguez, José Manuel Moyano, Ferran Ollé and Eric Teruel. "Estudio de técnicas de Inteligencia Artificial aplicadas a una plataforma de servicios móviles para la planificación de recursos móviles." no. 03/2009, Barcelona, IIIA-CSIC, 2009.

[13] Introduction to Genetic Algorithms Site. [Last search: 16/05/2012]. Available: http://www.obitko.com/tutorials/genetic-algorithms/

[14] Jaga - Java API for Genetic Algorithms Site. [Last search: 16/05/2012]. Available: http://www.jaga.org/

[15] Ismail Rakip Karas i Umit Atila. "A genetic algorithm approach for finding the shortest driving time on mobile devices." Scientific research and essays [1992-2248] Karas, 2011 vol.:6 iss:2 Pag.:394 -405

[16] Atsushi Maruyama, Naoki Shibata, Yoshihiro Murata, Keiichi Yasumoto, and Minoru Ito. 2004. "A Personal Tourism Navigation System to Support Traveling Multiple Destinations with Time Restrictions." In Proceedings of the 18th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA '04), Vol. 2. IEEE Computer Society, Washington, DC, USA, 18-.

[17] Yipeng Zhou, Junping Du, Feng Xu, Yang Yang, Xuyan Tu, "Tour Route Planning Based on Ant Colony System," Intelligent Systems, WRI Global Congress on, pp. 218-221, 2009 WRI Global Congress on Intelligent Systems, 2009

[18] Nearest neighbour algorithm. [Last search: 23/05/2012]. Available: http://en.wikipedia.org/wiki/Nearest_neighbour_algorithm

[19] Map Viewing and Navigation based on Open Street Maps for mobile devices [Last search: 03/06/2012]. Available: http://code.google.com/p/osmand

[20] Google Maps API [Last search: 03/06/2012]. Available: https://developers.google.com/maps/

[21] OSM XML [Last search: 03/06/2012]. Available: http://wiki.openstreetmap.org/wiki/OSM_XML

[22] Alastair Aitchison - Spatial stuff with SQL Server, Bing Maps, OSM and more [Last search: 03/06/2012]. Available: http://alastaira.wordpress.com/2011/02/09/loading-open-street-map-pois-with-sql-server-bing-maps/

[23] Fórmula del Haversine - Wikipedia [Last search: 06/06/2012]. Available: http://ca.wikipedia.org/wiki/F%C3%B3rmula_del_Haversine

[24] Algorisme de Dijkstra - Wikipedia [Last search: 06/06/2012]. Available: http://ca.wikipedia.org/wiki/Algorisme_de_Dijkstra
http://sourceforge.net/apps/mediawiki/travelingsales/

[25] Traveling Salesman [Last search: 06/06/2012]. Available: http://sourceforge.net/apps/mediawiki/travelingsales/

[26] OSM2PO - Routing on OpenStreetMap Data [Last search: 06/06/2012]. Available: http://osm2po.de/

[27] SLF4J Android [Last search: 06/06/2012]. Available: http://www.slf4j.org/android/