


UNIVERSITA' DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica (DIMES)

Dottorato di Ricerca in

Ingegneria dei Sistemi e Informatica

Con il contributo della Commissione Europea, Fondo Sociale Europeo e della Regione Calabria

CICLO

XXVII

Mobile Computing: Energy-Aware Techniques and Location-Based Methodologies

Settore Scientifico Disciplinare ING-INF/05

Coordinatore: Ch.mo Prof. Sergio Greco

Firma

Supervisore: Ch.mo Prof. Domenico Talia

Firma

Dottoranda: Dott.ssa Deborah Falcone

Firma

To my infinite love.

Preface

Mobile computing emerged and got a primary role thanks to the convergence of two technologies: the development of powerful portable computers and the development of wireless technology. Mobile computing represents a new paradigm of computing that is revolutionizing the way computers are used. It provides flexible communication among people and continuous access to networked services. The combination of networking and mobility has led to the emergence of new applications and services, such as collaborative computing to support disaster relief, health-care, construction systems, vehicle control, and navigation software to guide users in unfamiliar places and on tours.

However, the technical challenges in order to establish this paradigm of computing are not trivial. A main challenge faced by the software designers of a mobile computing system is ensuring energy efficiency, as most mobile devices are battery-power operated and lack a constant source of power.
Addressing the energy constraints of battery-operated wireless devices in decentralized environments is the first research objective described in this thesis. We propose an Energy-Aware (EA) scheduling strategy for allocating computational tasks over a network of mobile devices in a decentralized but effective way. Decentralization is achieved by defining a cooperative architecture in which mobile devices are clustered into local groups, named clusters.
To make the most of all available resources, the EA strategy tries to find a suitable distribution of tasks among clusters and individual devices. Specifically, the main design principle of our scheduling strategy is finding a task allocation that prolongs the total lifetime of a mobile network and maximizes the number of alive devices by balancing the energy load among them. To this end, the EA scheduler implements a two-phase heuristic-based algorithm.
We characterize the energy consumption of mobile devices defining an energy model in which the energy costs of both computation and communication are taken into account.
An extensive experimental evaluation has been performed to assess the performance of the proposed EA strategy in different network and application

scenarios. The experimental results show that by using the proposed energy-aware task allocation approach, the network lifetime is extended and the number of alive devices is significantly higher compared to related scheduling strategies, while meeting application-level performance constraints.

The rapid entrance to the mobile web era has also been marked by the rise of online social networks that explicitly use location as an essential element of their services. In fact, social media like Foursquare, Facebook and Twitter allow users to geo-tag the posts. The availability of large amounts of geographical and social data provides an unprecedented opportunity to study human mobile behavior through data analysis in a spatial-temporal-social context, enabling a variety of location-based services, that can support the decisions of city managers in transport planning, intelligent traffic management, route recommendations, etc.

The time and geo-coordinates associated with a sequence of social posts manifest the spatial-temporal movements of people in real life. The second research activity aims to analyze such movements to discover people and community behavior. With the information mentioned above, individuals can understand an unfamiliar city in a very short period and plan their journeys with minimal effort.

We worked on a methodology to extract spatial-temporal patterns from geo-tagged social post, and a statistical approach to build a real-life dataset of frequent travel sequences and interesting locations from this unpredictable and irregular information.

The proposed methodology consists of various phases. The first step is the collection of geo-tagged data from a social network. Then, we identify places of the considered urban area and associate to them a semantic label. Afterwords, we extract a set of daily trajectories and we used a sequential pattern mining algorithm to discover frequent travel routes. A key point is the definition of a set of spatial-temporal features over such routes and, accordingly, the statistical characterization of patterns, rules and regularities in moving trajectories. Such methodology helps in the automatic characterization of the dynamics of the urban environment, and could be integrated in location-based services useful for urban planning and management.

Prefazione

Il mobile computing è diventato una realtà grazie alla convergenza di due tecnologie: la realizzazione di potenti computer portatili e lo sviluppo della tecnologia wireless. Il mobile computing costituisce un nuovo paradigma computazionale che sta rivoluzionando il modo in cui vengono utilizzati i computer. Fornisce comunicazione flessibile tra le persone e l'accesso continuo ai servizi di rete. La combinazione di reti e mobilità ha portato alla nascita di nuove applicazioni e servizi, come il collaborative computing a supporto di catastrofi naturali, assistenza sanitaria, sistemi di costruzioni, controllo dei veicoli, e i sistemi di posizionamento e navigazione per guidare gli utenti in luoghi sconosciuti e in viaggio.

Tuttavia, le sfide tecniche al fine di stabilire questo paradigma non sono banali. La sfida principale, che deve essere affrontata dai progettisti software in un sistema di mobile computing è assicurare l'efficienza energetica, poichè la maggior parte dei dispositivi mobili è alimentata a batteria e non ha una fonte costante di potenza.
Il primo obiettivo di ricerca descritto in questa tesi è quello di affrontare i vincoli energetici dei dispositivi wireless a batteria in ambienti decentralizzati. A tal fine proponiamo una strategia di scheduling Energy-Aware (EA) per l'assegnazione di task computazionali su una rete di dispositivi mobili in modo decentrato ma efficace. Il decentramento è ottenuto definendo un'architettura cooperativa in cui i dispositivi mobili sono raggruppati in gruppi locali, denominati cluster.
Per sfruttare al meglio tutte le risorse disponibili, la strategia EA cerca una distribuzione ottimale dei task tra i cluster e i singoli dispositivi. Specificamente, il principio di progettazione della nostra strategia di scheduling è trovare l'allocazione dei task che prolunga la durata totale del tempo di vita della rete mobile e massimizza il numero di dispositivi accesi bilanciando il carico energetico tra loro. Per questa ragione, lo scheduler EA implementa un algoritmo euristico a due fasi.
Abbiamo caratterizzato il consumo energetico dei dispositivi mobili definendo un modello di energia in cui sono tenuti in considerazione sia i costi energetici

di computazione sia quelli di comunicazione.
Per valutare le prestazioni della strategia proposta in diversi scenari applicativi e di rete, è stata effettuata una vasta valutazione sperimentale. I risultati sperimentali mostrano che utilizzando l'approccio di scheduling energy-aware proposto, la durata di vita della rete è estesa e il numero di dispositivi accesi è significativamente più alto rispetto a quello di altre strategie di scheduling con cui EA è stato confrontato, nel rispetto dei vincoli di performance delle applicazioni.

La rapida entrata nell'era del mobile web è stata anche segnata dalla nascita delle reti sociali online che utilizzano esplicitamente la posizione geografica come elemento essenziale dei loro servizi. Infatti, social media come Foursquare, Facebook e Twitter consentono agli utenti di geo-referenziare i posti. La disponibilità di grandi quantità di dati sociali geo-referenziati offre un'opportunità senza precedenti di studiare il comportamento umano attraverso l'analisi dei dati in un contesto spazio-temporale-sociale, consentendo una varietà di servizi location-based, utili per supportare le decisioni di city manager nella pianificazione dei trasporti, nella gestione intelligente del traffico, nella raccomandazione di percorsi, ecc.
Le coordinate geografiche e temporali associate ad una sequenza di dati sociali manifestano i movimenti spazio-temporali delle persone nella vita reale. La seconda attività di ricerca mira ad analizzare tali movimenti per scoprire i comportamenti degli individui e della comunità. Utilizzando queste informazioni, le persone possono conoscere informazioni relative a una città sconosciuta in tempi brevi e pianificare i loro viaggi con il minimo sforzo.
La tesi presenta una metodologia per estrarre modelli spazio-temporali dai contenuti sociali geo-referenziati, e un approccio statistico per costruire un dataset reale di sequenze di viaggio frequenti e luoghi interessanti a partire da queste informazioni imprevedibili e irregolari.
La metodologia proposta si compone di varie fasi. Il primo passo è la raccolta dei dati geo-referenziati da una social network. Successivamente, identifichiamo i luoghi della zona urbana considerata e associamo ad essi un'etichetta semantica. In seguito, sono estratte un insieme di traiettorie quotidiane e viene usato un algoritmo di sequential patter mining per scoprire itinerari di viaggio frequenti. Un punto chiave è la definizione di un insieme di caratteristiche spazio-temporali su tali rotte e, sulla base di queste, la caratterizzazione statistica di modelli e regolarità nelle traiettorie di movimento estratte. Tale metodologia aiuta la caratterizzazione automatica delle dinamiche dell'ambiente urbano, e potrebbe essere integrata in servizi basati sulla localizzazione utili per la pianificazione e gestione urbana.

Arcavacata di Rende, Cosenza, Italy *Deborah Falcone*
November 2014

Contents

1

Introduction

The emergence of powerful portable computers, along with advances in wireless communication technologies, has made mobile computing a reality.
Mobile devices (e.g., smartphones, tablets, etc) are increasingly becoming an essential part of human life as the most effective and convenient communication tools not bounded by time and place. Mobile users accumulate rich experience of various services from mobile applications (e.g., iPhone apps, Google apps, etc), which run on the devices and/or on remote servers or clouds via wireless networks.

Mobile computing is a versatile and potentially strategic technology that improves information quality and accessibility, increases operational efficiency, and enhances management effectiveness. Its rapid progress becomes a powerful trend in the development of IT technology as well as commerce and industry fields [8]. This has opened the way to the spreading of a huge number of mobile web-based services and location-aware services in many real-life mobile scenarios, such as body-health monitoring, vehicle control, wireless security systems, risk management.

Having billions of mobile users and smart devices, mobile services and digital data will grow explosively. However, mobile computing must face many challenges such as limited availability of battery power, low-bandwidth networks, relatively small storage space, mobility, and security in a wireless environment [7]. The limited resources significantly impede the improvement of service qualities.

Energy seems to be a key and not completely solved issue with these devices. Screens are getting bigger and more colorful, functionality is increasing, as is the number of network interfaces and sensors available and used by applications. Several of the most recent smart phones last no more than 12 hours without recharging, when continuously connecting to the network. While some of these issues will be solved at the hardware level by using more sophisticated batteries or energy-aware solutions, there is also a need for improved software solutions that work effectively [9].

One important aspect to note about the recent use of mobile computing is the increased access to social networking services from mobile devices, that are quickly becoming the most popular platform for users to create and upload data in many existing social computing applications.
One of the most popular trends in social networking is geolocation technology. This technology uses data acquired from mobile devices to identify or describe the actual physical location of the users, giving the ability to exploit people and community behavior.
For example, photo-sharing site Flickr has enjoyed more than 50% increase in traffic on its mobile site and over 100 million photos are geo-tagged. By leveraging the vast amount of such geo-tagged multimedia data, researchers at Google developed technologies to automatically recognize landmarks that are popular tourist destinations and build their visual models from the photos. The mobile social networking site Foursquare allows users to share location and status with their friends and earn credits via sharing valuable information about places. Rather than just being social networking tool, many shops and restaurants recognize Foursquare's value as a marketing platform and provide benefits such as free drinks or discounts to highly active users as a way to encourage visits to their venues. The micro blogging service Twitter also heavily relies on the SMS services on mobile phones for users to conveniently send and receive status updates whenever and wherever they want, which is the key feature that distinguishes it from traditional blogging services and instant messengers. An important implication of the success of Twitter like applications is that they provide strong evidence that people have a strong tendency toward sharing more and more personal data covering various aspects of daily life[1]. Moreover, the ability to associate spatial context to posts is becoming a popular feature of Twitter that exploits the GPS readings of users' phones to tag posts, photos and videos with geographical coordinates.

Given the mobile scenario discussed so far, the main aim of this thesis is designing techniques and methodologies that allow professionals to develop mobile services by optimally utilizing the limited resources and take advantage of the extensive amount of available data to aid the automatic characterization of the dynamics of the urban environment.

1.1 Objectives of the Research

1.1.1 Mobile Energy Efficiency

The continuous advances in mobile computing technologies have opened the way to the spreading of mobile applications. In spite of this, the limited battery-power of mobile devices still prevents their wider use.
To make the most of all available resources, we focus on the problem of ensuring energy efficiency to meet the needs of energy-hungry modern applications

[1] http://www.huawei.com/en/

and to enable effective and reliable computing over mobile devices that collaborate each other.
The first research activity described in this thesis aims at filling the lack of task allocation strategies in mobile environments, addressing both the energy constraints of mobile devices and the decentralized nature of wireless networks. We focus on this two-fold issue by proposing an energy-aware (EA) scheduling strategy for allocating computational tasks over a network of mobile devices in a decentralized way.
We designed a distributed mobile-to-mobile architecture, in which mobile devices cooperate according to a peer-to-peer style, and an energy-aware model to characterize the energy consumption of mobile devices. Within this framework, the EA scheduling strategy works to reduce energy consumption of the mobile device that uses it. The main principle at the basis of our solution is that whenever a resource-limited mobile device has a task to execute, that task can be assigned to another mobile device that can handle its execution more effectively. The EA scheduler implements a two-phase heuristic based on a decentralized approach. The results show that EA finds a task allocation that prolongs the total lifetime of a mobile network and maximizes the number of alive devices by balancing the energy load among them.

1.1.2 Mobile Social Analysis

Another research activity described in this thesis was carried out for managing the explosive mobile data volume coming from mobile devices and posted on social networks.
Many of devices combine the features of mobile phone with GPS navigation units. This combination favored the affirmation of the location based social networks (LBSNs), such as Foursquare. Moreover, many of the most popular social networks, like Facebook and Twitter, have started allowing users to tag their posts with geographical coordinates.
The ability to associate spatial context to on-line posts is allowing to collect a huge amount of social data, which although collected on line, incorporates information about the actions of people in real life.
Our purpose is to define and implement innovative methodology to discover people and community behavior, i.e. patterns, rules and regularities in moving trajectories. To a better understanding of users' patterns we enrich latitude-longitude coordinates with semantic information. Knowing the semantics of the type of a place (home, office, museum) a user is at is potentially very useful. It could allow to infer users common interests, to improve activity prediction and ultimately mobile user recommendation and advertisement.
The proposed methodology consists of various phases. In the first step geo-tagged data are collected from a social network. Due to the variable accuracy of GPS, it is necessary to cluster the locations of the geo-tagged data so that each place is identified by a single pair of geographic coordinates and associate to them a semantic label. Then, we extracted a set of daily trajectories and we

used a sequential pattern mining algorithm to discover frequent travel routes. If a trajectory pattern repeats frequently, we consider it as a frequent travel route. Finally, we extract spatial-temporal information for each of them to capture the factors that may drive users movements.
This methodology can be integrated in a recommender system for trip planning, personalized navigation services, and location-based services useful for urban planning and management.

1.2 Publications

The following publications have been produced while accomplishing this thesis.

1.2.1 Journal

- C. Comito, D. Falcone, D. Talia, P. Trunfio, "Efficient Allocation of Data Mining Tasks in Mobile Environments". Concurrent Engineering: Research and Applications, vol. 21, n. 3, pp. 197–207, September 2013.

1.2.2 Papers in refereed conference proceedings

- D. Falcone, C. Mascolo, C. Comito, D. Talia, J. Crowcroft, "What is this place? Inferring place categories through user patterns identification in geo-tagged tweets". Proc. of the 6th International Conference on Mobile Computing, Applications and Services (MobiCASE 2014), November 2014.
- D. Falcone, "Energy-aware techniques and location-based methodologies in mobile environments". International Conference on High Performance Computing & Simulation (HPCS), pp.991–994, July 2014.
- C. Comito, D. Falcone, D. Talia, P. Trunfio, "A distributed allocation strategy for data mining tasks in mobile environments". Proc. of the 6th International Symposium on Intelligent Distributed Computing (IDC 2012), Calabria, Italy, Studies in Computational Intelligence, vol. 446, pp. 231–240, Springer, September 2012.
- C. Comito, D. Falcone, D. Talia, P. Trunfio, "Energy Efficient Task Allocation over Mobile Networks". Proc. of the 9th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2011), Sydney, Australia, pp. 380–387, IEEE Computer Society Press, December 2011.

1.2.3 Other publications

- C. Comito, D. Falcone, D. Talia, P. Trunfio, "Scheduling Data Mining Applications in Mobile Computing Environments". ERCIM News, n. 93, pp. 15–16, April 2013.

- C. Comito, D. Falcone, D. Talia, "Mining Popular Travel Routes from Social Network geo-tagged data. *Submitted*.
- C. Comito, D. Falcone, D. Talia, P. Trunfio, "Energy-Aware Task Allocation in Mobile Environments". *Submitted*.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows: Chapter 2 presents the Mobile Computing concepts, outlining the main features, benefits, challenges, applications and the role in social era; Chapter 3 focuses on the proposed energy-aware scheduling strategy, presenting a short description of background and existing work, the architecture of our framework, the energy model, and two different implementations of the system; in Chapter 4 is discussed a novel methodology to discover human dynamics in an urban area through the analysis of geo-tagged social data, each phase of our approach is detailed, emphasizing the complex aspect of automatic semantics association to geographical locations; finally, in Chapter 5 conclusions and future work are discussed.

2

Mobile Computing Concepts

Mobile computing refers to computing elements and services that can be moved during their usage. Mobile computing is therefore indivisibly related with the miniaturization of technology, connectivity, communication, and storage facilities that extend the capabilities of lightweight mobile devices. The combined use of these technologies on personal devices enables people to access their personal information as well as public resources *anytime* and *anywhere*.

Portable and small computers such as laptop computers, smartphones, personal digital assistants, smartcards, watches and the like, are gaining wide popularity. Their computing capabilities are growing quickly, while they are becoming smaller and ubiquitous. Such advances have enabled computing to become location-independent, marking a gradual transition from room-filling mainframes, personal computers, and laptops to truly mobile devices that are now ever-present and taken for granted [105]. Mobile computing enables the localization and personalization of content, always providing the user with the right information at the right time [106].

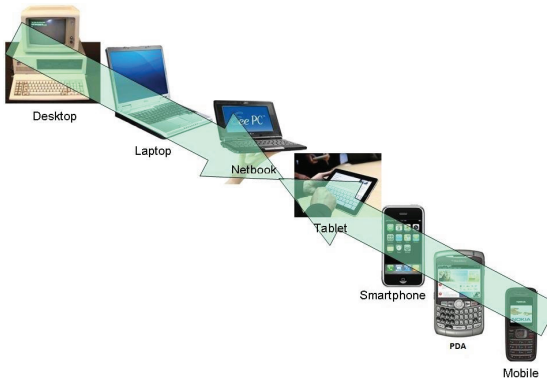


Fig. 2.1. Evolution of computers

One of the things that makes mobile computing an interesting topic of research and design is that the area is strongly driven by innovation, characterized by rapidly evolving use, and has enormous market potential and growth. New technologies are constantly being developed, new use domains are constantly being explored, and successful new ideas and applications reach millions of users. In fact, by the end of 2010 more smartphones than personal computers were, for the first time, being sold worldwide, reflecting this dynamic and rapidly evolving nature of the mobile area.
The uptake of mobile technology in our work and private spheres has had a huge impact on the way we perceive and use these technologies. They are no longer just computers on batteries. They have become functional design objects, the look, feel and experience of which we care deeply about, and that we juggle in multitude in our everyday lives.

2.1 Characteristics

Mobile computing is made possible by portable computer hardware, software, and communications systems that interact with non-mobile information systems while away from the normal, fixed workplace. Mobile computing is a versatile and potentially strategic technology that improves information quality and accessibility, and offer the following properties:

- *Portability.* The ability to move a device within an environment or to different environments with ease.
- *Social Interactivity.* The ability to share data and collaboration between users.
- *Connectivity.* The ability to be digitally connected for the purpose of communication of data in any environment.
- *Location flexibility.* The ability to enable user to work from anywhere as long as there is a connection established. A user can work without being in a fixed position. Their mobility ensures that they are able to carry out numerous tasks at the same time perform their stated jobs.
- *Saves Time.* The ability to slash the time consumed or wasted by traveling from different locations or to the office and back. One can now access all the important documents and files over a secure channel or portal and work as if they were on their computer. It has enhanced telecommuting in many companies. This also reduces unnecessary expenses that might be incurred. Meetings, seminars and other informative services can be carried out using the video and voice conferencing. This cuts down on travel time and expenditure.
- *Entertainment.* The ability to access easily a wide variety of movies, educational and informative material. Video and audio recordings can be streamed on the go using mobile computing. With the improvement and availability of high speed data connections at considerable costs, one is

able to get all the entertainment they want as they browser the internet for streamed data. One can be able to watch news, movies, and documentaries among other entertainment offers over the internet. This was not such before mobile computing dawned on the computing world.

- *Streamlining of Business Processes.* The ability to easily make available the business processes through secured connections. Based on the factor of security, adequate measures have been put in place to ensure authentication and authorization of the user accessing those services.

Mobile computing [6] is accomplished using a combination of computer hardware, system and applications software and some form of communications medium. Powerful mobile solutions have recently made possible because of the availability of an extremely powerful and small computing devices, specialized software and improved telecommunication.
A brief overview of the different types of hardware, software, and communications mediums that are commonly integrated to create mobile computing solutions are defined.

Hardware. In mobile computing, to be portable, devices must be small, light and operational under wide environmental conditions. Also, in the context of ubiquitous or pervasive computing, computational power is embedded in numerous small devices. In particular, mobile devices have small screens and small multifunction keypads, this necessitates the development of appropriate user interfaces. Those devices have less resources than static elements, including memory, disk capacity and computational power than traditional computing devices. Portable devices rely for their operation on the finite energy provided by batteries. Even with advances in battery technology, this energy concern will not cease to exist. The concern for power consumption spans various levels in hardware and software design.

System software. Mobile computers make use of a wide variety of low-level and application software. The most common operating systems used on mobile computers include Android, iOS, Windows Phone, Embedded Linux, etc. A mobile operating system, also referred to as mobile OS, combines the features of a personal computer operating systems with other features, including a touchscreen, Bluetooth, Wi-Fi, GPS mobile navigation, camera, speech recognition, voice recorder, music player, near field communication services and infrared blaster.
Each operating system/environment has some form of integrated development environment (IDE) for application development. Most of the operating environments provide more than one development environment option for custom application development.
Android is the most popular mobile OS. Some of the most famous smartphone OS, like Symbian, was overtaken by Android at the and of 2010. As of 2013, Android devices sell more than Windows, iOS, and Mac OS devices combined, as evidenced by the trend of smartphone sales from 2007 to 2013, shown in

Figure 2.2[1]. In 2014, more than a billion smartphones were sold and global market share was 85% for Android, 11% for iOS, 3% for Windows Phone and less than 1% for all other platforms[2].

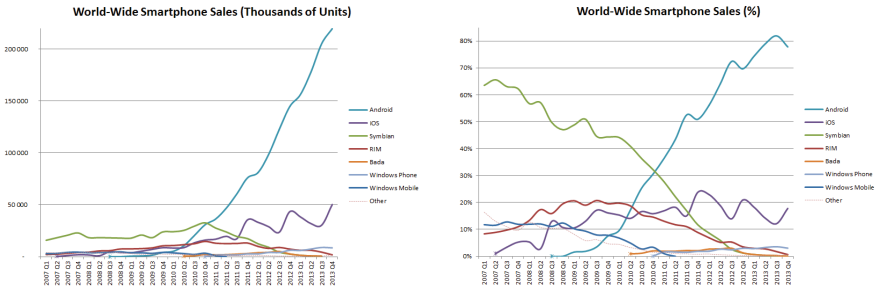


Fig. 2.2. World Wide Smartphone Sales in terms of thousands of units (a) and percentage values (b).

Communication. The ability of a mobile computer to communicate in some way with a fixed information system is a defining characteristic of mobile computing. The type and availability of communication medium significantly impacts the type of mobile computing application that can be created. The way a mobile computing device communicates with a fixed information system can be categorized as: (a) connected (b) weakly connected (c) batch and (d) disconnected. The connected category implies a continuously available high-speed connection. The ability to communicate continuously, but at slow speed, allows mobile computers to be weakly connected to the fixed information system. A batch connection means that the mobile computer is not continuously available for communication with the fixed information system. In the batch mode, communication is established randomly or periodically to exchange and update information between a mobile device and fixed information systems. Mobile computers may operate in batch mode over communication mediums that are capable of continuous operation, reducing the wireless airtime and associated fees.

Disconnected mobile computers allow users to improve efficiency by doing calculations, storing contact information, keeping a schedule, and other non-communications oriented tasks. This mode of operation is of little interest because the mobile device is incapable of interacting and exchanging information with a fixed organizational information system. Exchange of information with a disconnected mobile computing device can only be accomplished by manually entering information into the device or copying from the device's

[1] en.wikipedia.org
[2] www.huffingtonpost.co.uk

screen and manually entering the information into the fixed information system. This mode of information exchange is not really efficient with respect to using paper and is effectively not used, since virtually all modern mobile computing hardware is capable of native electronic data communications and sharing.
Data Communications is the exchange of data using existing communication networks. The term data covers a wide range of applications including File Transfer, interconnection between Wide-Area-Networks (WAN), facsimile (fax), electronic mail, access to the internet and the World Wide Web (WWW).
There are many communications technologies available today that enable mobile computers to communicate. The most common of these technologies are: Wireless Local Area Networks (WLANs); Satellite; Cellular Digital Packet Data (CDPD); Personal Communications Systems (PCS); Global System for Mobile communications (GSM); RAM and ARDIS data networks; Specialized Mobile Radio (SMR) service; one and two-way paging; plain old telephone system (POTS); Internet; infra-red; docking (serial, parallel, LAN); and disk swapping [107].

2.2 Middleware

Many research work has been done in the past years for porting traditional distributed systems middleware to mobile and pervasive distributed systems. However, mobility is increasingly meaningful, thus a new core of abstractions that extend distributed middleware with support to mobility must be fully researched.
Mobile devices require light computational load, thus existing middleware for heavy computational load such as that found in traditional distributed systems cannot be used. The intermittent connection nature of mobile systems also requires an asynchronous form of communication. Additionally, unlike fixed distributed systems, mobile systems execute in an extremely dynamic context which in turn necessitates that devices must be aware continuosly of their environment.

There are examples of use of traditional middleware systems in the context of mobile computing, such as the adaptation of *object-oriented middleware* and *message oriented middleware* to small and mobile devices. The main problem with the object-oriented approach is that it relies on synchronous communication primitives that do not necessarily suit all the possible mobile system architectures. The computational load of these systems is quite high and the principle of transparency they adhere to does not always fit mobile applications. Oracle provides *J2ME* (Java Micro Edition) which is a basic JVM and development package targeting mobile devices. Microsoft recently matched this with *.Net* Compact Framework, which also offers support for XML data and web services connectivity. However, this approach is not a real

solution, as it completely relies on application designers for the solution of most of the non-functional requirements middleware should provide, starting from scalability [108].

On the other hand, recently, some middleware specifically targeting the needs of mobile computing have been devised. The proposed middleware for mobile distributed systems is split into different research areas such as context-aware middleware, data sharing-oriented middleware and tuple space-based middleware.

2.2.1 Context-Awareness Based Middleware

To enable applications to adapt to heterogeneity of hosts and networks as well as variations in the user's environment, systems must assure that mobile applications are aware of the context in which they are being used. Furthermore, context information can be used to optimize application behavior counter balancing the scarce resource availability. User's context includes, but is not limited to:

- *location*, with varying accuracy depending on the positioning system used;
- *relative location*, such as proximity to printers and databases;
- *device characteristics*, such as processing power and input devices;
- *physical environment*, such as noise level and bandwidth;
- *user's activity*, such as driving a car or sitting in a lecture theater.

The principle of *Reflection* has often been used to allow dynamic reconfiguration of middleware and has proven useful to offer context-awareness. The concept of reflection was first introduced by Smith in 1982 [124] as a principle that allows a program to access, reason about and alter its own interpretation. Context information can be kept by middleware in its internal data structures and, through reflective mechanisms, applications can acquire information about their execution context and tune the middleware behavior accordingly. No specific communication paradigm is related to the principle of reflection, so this issue is left unspecified and depends on the specific middleware system built. Some recent approaches have investigated the use of reflection in the context of mobile systems, and used it to offer dynamic context-awareness and adaptation mechanisms.
Example of context-aware system is *UIC*(Universally Interoperable Core) and *Gaia* [108].

Location-aware Middleware

Location is one of the most studied aspects of context awareness. Location awareness has attracted a great deal of attention and many examples exist of applications that exploit location information to offer travelers directional guidance; find out neighboring devices and the services they provide; send advertisements depending on user's location; send messages to anyone in a

specific area; and so on.
To enhance the development of location-based services and applications, and reduce their development cycle, middleware systems have been built that integrate different positioning technologies by providing a common interface to the different positioning systems (e.g., GPS and infrared). Examples include *Oracle iASWE*, *Nexus*, *Alternis*, *SignalSoft*, *CellPoint*, and many others are being released [108].

2.2.2 Data Sharing-Oriented Middleware

One of the major issues targeted is the support for disconnected operations and data sharing. Systems like *Coda*, its successor *Odyssey*, *Bayou* and *Xmiddle* try to maximize availability of data, giving users access to replicas. They differ in the way they ensure that replicas move towards eventual consistency, that is, in the mechanisms they provide to detect and resolve conflicts that naturally arise in mobile systems. Despite a proliferation of different, proprietary data synchronization protocols for mobile devices, we still lack a single synchronization standard, as most of these protocols are implemented only on a subset of devices and are able to access a small set of networked data. This represents a limitation for both end users, application developers, service providers and device manufacturers [108].

2.2.3 Tuple Space-Based Middleware

The characteristics of wireless communication media (e.g., low and variable bandwidth, frequent disconnections, etc.) favor a decoupled and opportunistic style of communication: decoupled in the sense that computation proceeds even in presence of disconnections, and opportunistic as it exploits connectivity whenever it becomes available. The synchronous communication paradigm supported by many traditional distributed systems has to be replaced by a new asynchronous communication style. Tuple space systems have been shown to provide many useful facilities for communication in wireless settings.
Tuples constitute the basic elements of a tuple space systems; they can be seen as vector of typed values. Tuples are created by a process and placed in the tuple space using a *write* primitive, and they can be accessed concurrently by several processes using *read* and *take* primitives, both of which are blocking (even if non-blocking versions can be provided).

Tuples are anonymous, thus their selection takes place through pattern matching on the tuple contents. Communications is de-coupled in both time and space: senders and receivers do not need to be available at the same time, because tuples have their own life span, independent of the process that generated them, and mutual knowledge of their location is not necessary for data exchange, as the tuple space looks like a globally shared data space, regardless of machine or platform boundaries. These forms of decoupling assume enormous importance in a mobile setting, where the parties involved

in communication change dynamically due to their migration or connectivity patterns. However, a traditional tuple space implementation is not enough. There are basic questions that need to be answered: how is the globally shared data space presented to mobile hosts? How is it made persistent? The solutions developed to date basically differ depending on the answers they give to the above questions. Three tuple-space middleware that have been proposed for mobile computing applications are *Lime*, *TSpaces* and *L2imbo* [108].

2.3 Fundamental Challenges

In this section we examine the impact of the principal features of Mobile Computing: wireless communication, mobility and portability. Wireless communication brings challenging network condition; mobility causes greater dynamicism of information; and portability entails limited resources available on board to handle the changeable mobile computing environment.
Mobile computing is characterized by four constraints [7].

Mobile elements are resource-poor relative to static elements

For a given cost and level of technology, considerations of weight, power, size and ergonomics will exact a penalty in computational resources such as processor speed, memory size, and disk capacity. While mobile elements will improve in absolute ability, they will always be resource-poor relative to static elements. For instance, screens and keyboards tend to be small, which may make them hard to use. Moreover, alternate input methods such as speech or handwriting recognition require training.

Mobility is inherently hazardous

When working mobile, one is dependent on public networks, requiring careful use of Virtual Private Network (VPN). Security is a major concern while concerning the mobile computing standards on the fleet. One can easily attack the VPN through a huge number of networks interconnected through the line. There are various kinds of security issues [107]:

- *Confidentiality.* Preventing unauthorized users from gaining access to critical information of any particular user.
- *Integrity.* Ensures unauthorized modification, destruction or creation of information cannot take place.
- *Availability.* Ensuring authorized users getting the access they require.
- *Legitimate.* Ensuring that only authorized users have access to services.
- *Accountability.* Ensuring that the users are held responsible for there security related activities by arranging the user and his/her activities are linked if and when necessary.

In addition to security concerns, portable computers are more vulnerable to loss or damage.

Mobile connectivity is highly variable in performance and reliability

Mobile Internet access is generally slower than direct cable connections, using technologies such as GPRS and EDGE, and more recently 3G networks. These networks are usually available within range of commercial cell phone towers. Higher speed wireless LANs are inexpensive but have very limited range.
As the mobile computers move they encounter networks with different features. Some buildings may offer reliable, high-bandwidth wireless connectivity while others may only offer low-bandwidth connectivity. Outdoors, a mobile client may have to rely on a low-bandwidth wireless network with gaps in coverage.
Moreover a mobile computer must be able to switch from infrared mode to radio mode as it moves from indoors to outdoors. Additionally it should be capable of switching from cellular mode of operation to satellite mode as the computer moves from urban and rural areas. In mobile computing as computers are working in cells and are being serviced by different network providers, the physical distance may not reflect the true network distance. A small movement may result in a much longer path if cell or network boundaries are crossed. This can increase the network latency as well as risk of disconnection. Service connections must be dynamically transferred to the nearest server. However, when load balancing is a priority this may not be possible [107].

Mobile elements rely on a finite energy source

Mobile Computers will rely on their batteries as the primary power source. While battery technology will undoubtedly improve over time, the need to be sensitive to power consumption will not diminish. Concern for power consumption must span many levels of hardware and software to be fully effective. Batteries should be ideally as light as possible but at the same time they should be capable of longer operation times. Power consumption should be minimized to increase battery life.
Energy efficiency is a key concern to enable effective and reliable computing over mobile devices. We worked in this direction by defining a distributed architecture in which mobile devices cooperate in a peer-to-peer style to perform tasks, tackling the problem of energy capacity shortage by distributing the energy consumption among the available devices.

2.4 Applications

Mobile computing is using in a large variety of application areas [122] and every day new applications of mobile computing systems appear. Here we mention some significant application domains of mobile computing systems.

Emergency Services. Ability to receive information on the move is vital where the emergency services are involved. Information regarding the address, type and other details of an incident can be dispatched quickly, using mobile computers, to one or several appropriate mobile units, which are in the vicinity of the incident.

In courts. Defense counsels can take mobile computers in court. When the opposing counsel references a case which they are not familiar, they can use the computer to get direct, real-time access to on-line legal database services, where they can gather information on the case and related precedents. Therefore mobile computers allow immediate access to a wealth of information, making people better informed and prepared.

In companies. Managers can use mobile computers in presentations to major customers. They can access the latest market share information. At a small recess, they can revise the presentation to take advantage of this information. They can communicate with the office about possible new offers and call meetings for discussing responds to the new proposals. Therefore, mobile computers can leverage competitive advantages.

Credit Card Verification. At Point of Sale (POS) terminals in shops and supermarkets, when customers use credit cards for transactions, the intercommunication is required between the bank central computer and the POS terminal, in order to effect verification of the card usage, can take place quickly and securely over cellular channels using a mobile computer unit. This can speed up the transaction process and relieve congestion at the POS terminals.

Field Sales. The operational efficiency of sales personnel is significantly enhanced through mobile computing. An excellent example of these improvements can be seen by examining how mobile computing improves the efficiency of remote insurance and financial planning sales. The mobile computer frees the sales agent to meet with the client at the client's home, office, or other location. Customer data is collected, estimates and comparisons are immediately calculated, the customer decides on the program of choice, the central computer is immediately updated, and the customer is enrolled in the insurance or financial planning program. Without mobile computing, this sales process would take days instead of minutes. In addition to accessing and updating customer account information, mobile sales personnel can accomplish tasks such as printing invoices or other information to leave with the customer.

Transportation and Shipping. Using mobile computers in conjunction with GPS/GIS and an accompanying vehicle information system (VIS), the operations of an entire transportation fleet can be managed from a central location. The central office knows the location, status, and condition of all vehicles, and operators have two-way communication with the operations center. Using this information, vehicles can be optimally dispatched to maximize efficiency as measured by: (a) time; (b) fuel consumption; and (c) delivery priority. The mobile computers enable significant performance improvements, achieved simultaneously with operational cost reductions.

General Dispatching. Mobile computers used in conjunction with Global Positioning System (GPS) and Geographical Information System (GIS) data allow significant improvements in the operational efficiency of various dispatch operations. For example, the central computer of a taxi company can track the location and status of all its taxicabs and electronically dispatch the most appropriate car to a customer's location. Additionally, the central computer can calculate the arrival time of the taxi, enabling improved customer service.

Hotel Operations. Connecting the cleaning and hospitality staff of a hotel with mobile computing can significantly improve the efficiency of hotel operations. As guests check out and rooms are vacated, the central computer wirelessly signals cleaning staff that the rooms are ready for cleaning. Problems that are identified during cleaning, such as broken appliances or faulty plumbing, are immediately communicated to the mobile maintenance team for action. As soon as cleaning is complete and repairs are accomplished, the cleaning staff member wirelessly updates the central computer and the room is immediately available for check-in by a new guest. The same system can be used to efficiently direct mobile hospitality personnel in response to guest requests for information and service.

News Reporting. Mobile computers dramatically improve the efficiency of news media operations. Reporters equipped with mobile computers and accompanying electronic devices can cover news or sporting event, take digital video or still photographs, digitally record audio interviews, compose the text of the news story, and transmit the completed product back to the central agency for editing and immediate publication. In the media industry, the timing and quality of news coverage is critical. Mobile computing increases the quality of the information from the media crews and significantly decreases the time required to process and transmits the story for publication.

Health Care. Mobile medical care, whether in-home, on the road, or within a hospital, is more efficient with mobile computing. The mobile health care worker can access patient records for reference purposes, and is able to update records with current diagnosis and treatment information. Emergency medical technicians (EMTs) responding at the scene of an accident can use mobile computers to capture patient information, treatments accomplished vital signs, and other critical data. This information is wirelessly transmitted to the receiving hospital, which then prepares to receive and treat the patient, or recommend another hospital facility with more appropriate treatment facilities depending upon the nature and severity of the injuries. The more efficient hand-off between ambulance EMTs and hospital staff made possible by mobile computing can save lives that otherwise might have been lost.

Fieldwork. Almost any form of fieldwork can be made significantly more efficient through the application of mobile computing. Parking control officers and utility inspectors are two examples of field workers who can receive operational benefits from mobile computing. Parking control officers use handheld

computers to check the registration and violation history of parking offenders. Parking violations are issued immediately and towing/backup can be requested when required. Utility inspectors have historically used paper forms to capture information such as consumer power consumption and utility equipment status (transformers, transmission towers, etc.). Using mobile computers, inspectors can be given instructions on inspections to be accomplished and information can be captured and validated at the source.

Mobile Automation. General business travelers also reap the benefits of mobile computing. E-mail, spreadsheets, presentations, and word processing are the four primary tasks accomplished by these business travelers. Laptops, palmtops, and portable computers with usable-size keyboards enable business people to stay in touch and accomplish the tasks they need for job effectiveness. Using powerful mobile computers in conjunction with high-speed connectivity, mobile workers can perform work normally accomplished at the office while on the road or in the field.

2.5 The Impact on Social Media

The impact of mobile devices in our modern world is getting everyday clearer and more important. With the impact that these devices have had on social media platforms, mobile computing just got a higher attention from hardware and software developers and vendors.
Because of the technological evolution in recent years, mobile devices are pervasively connected to global network as well as to persons. As a consequence of the development of location based services and the rapid adoption of mobile devices such as cell phones, gaming machines and handheld computers by a wide variety of users, social networks have indeed migrated onto the mobile platform [109].
The International Data Corporation (IDC) indicates that the most popular activities of mobile internet users are search engine querying, reading news information, downloading multimedia, using email and instant messaging clients, but the fastest-growing activities are social networking and blogging. In fact, the recent use of mobile computing is the increased access to social networking services from mobile devices. Mobile users spend more time on social networking sites than desktop users. These statistics have several implications for systems and network builders in general. The number of mobile phone users is already much higher than desktop users, with much more dynamic connectivity patterns [9].
This means that things like customer experience with low latency, high responsiveness and ease of interaction will need major improvements if customers are expected to have the same kind of experience on mobile platforms as desktop users. This has an impact, among other effects, on the load of systems handling contents for social networking sites.

The combination of mobile phone technologies and online social networks leads to a very appealing world for mobile advertisers and more generally for mobile service providers. Mobile advertisers will reap a huge chunk of that revenue as innovation in the field of geo-location, it is expected to lead to more targeted and therefore better advertising. Location-based social networks such as Foursquare are predicated on the use of this technology from a mobile device. Moreover, the launch of Facebook Places and the ability to add longitude and latitude to tweets in Twitter, are evidence that the future of mobile computing is in social media.

Part of this forecast relies on the advantages of accessing social network services from a mobile device: through location information gathering, sensing user activity as well as social profiling can become more targeted and therefore better received and useful.

This means that messages posted on the social networks are not only generated when users are at their desk but also at any time throughout the day. They also come from myriad devices dispersed and roaming all over the world. And this is not limited to traditional messages: items of interest increasingly include video download, music, and media streaming [9].

Unquestionably, mobile services and digital data will grow explosively. To handing the huge quantity of mobile data we design techniques and methodologies that allow to take advantage of the extensive amount of available data to aid the *automatic characterization of the urban environment dynamics.*

3

Energy Efficiency in Mobile Environments

Continuous advances in mobile computing technologies opened the way to the spreading of mobile applications. Today, the worldwide mobile communication infrastructure offers ubiquitous mobile connectivity and unprecedented service levels. Billion of users around the world depend on their mobile phones for their private and professional lives; however, this has come at the cost of increasing energy consumption. Mobile devices such as smart-phones, personal digital assistants (PDAs), and tablets are battery-power operated, and thus far, battery technologies do not meet the needs of modern energy-hungry applications.

We have focused on the problem of ensuring *energy efficiency* to enable effective and reliable computing over mobile devices. In particular, we have considered distributed applications running on Mobile ad-hoc networks (MANETs), that are self-configuring networks of mobile devices connected by ad-hoc wireless links, that collaborate each other in a peer-to-peer style [1]. Examples of mobile-to-mobile collaborations can be found in several context such as disaster relief, health-care, construction systems and vehicle control. The basic principle behind such cooperative scenarios is that, whenever a resource-limited mobile device has a task to execute, that task can be assigned to another mobile device that can handle its execution more effectively.

An issue that still prevents a wider implementation of distributed applications in mobile-to-mobile scenarios is the lack of task allocation strategies addressing both the energy constraints of battery-operated wireless devices and the decentralized nature of MANETs. We have tackled on this two-fold issue by proposing an energy-aware (EA) scheduling strategy for allocating computational tasks over a network of mobile devices in a decentralized but effective way. Decentralization is achieved by clustering mobile devices into local groups, named clusters. Such a cooperative architecture can be seen as a set of requestors, i.e., mobile applications generating tasks to be executed, and a clustered set of resources, i.e., mobile devices characterized by different levels of energy and processing power, where tasks can be executed. To make the most of all available resources, the EA strategy tries to find a suitable

distribution of tasks among clusters and individual devices. Specifically, the main design principle of our scheduling strategy is finding a task allocation that prolongs the total lifetime of a mobile network and maximizes the number of alive devices by balancing the energy load among them. To this end, the EA scheduler implements a two-phase heuristic-based algorithm. The algorithm first tries to assign a task locally to the cluster where the execution request has been generated, so as to maximize the cluster residual life. If the task cannot be assigned locally, the second phase of the algorithm is performed by assigning the task to the most suitable node all over the network of clusters, maximizing this way the overall network lifetime.
We characterize the energy consumption of mobile devices defining an energy model in which the energy costs of both computation and communication are taken into account. There are several works in literature whose goal is minimizing the overall energy dissipation of the system [14, 42]. However, this goal does not capture the nature of cooperative ad-hoc systems. The reason is that minimizing the overall energy dissipation can lead to heavy use of energy-effective devices, regardless of their remaining energy. The consequent short lifetime of such devices will very likely compromise the system performance. This weakness is a major motivation of the proposed energy-balanced task allocation scheme.
An extensive experimental evaluation has been performed to assess the performance of the proposed EA strategy in different network and application scenarios.

The remainder of this chapter is organized as follows. Section 3.1 provides a description of basic concepts and discusses related work. Section 3.2 presents the energy-aware architecture of the framework, describes the software components of mobile devices and the energy model, while in Section 3.3 is introduced the energy-aware scheduling scheme. Sections 3.4 and 3.5 discusses two different implementations of the system, respectively a prototype and a discrete-event simulator, and for both of them presents the performance evaluation of the proposed EA strategy. Finally, Section 3.6 concludes the chapter.

3.1 Background and Related Work

This section provides a background on the mobile energy-aware framework and discusses related work.

3.1.1 Mobile Ad-hoc Network

A network of mobile nodes connected in wireless in ad-hoc mode is called MANET, namely Mobile Ad-hoc NETwork. This kind of networks are an emerging area of mobile computing and ad-hoc networking capabilities are expected to become an important part of overall next-generation wireless

network functionalities. In general, mobile ad-hoc networks are formed dynamically by an autonomous system of mobile nodes (PDAs, mobile phones, laptops) that are connected via wireless links without using an existing network infrastructure or centralized administration. The nodes are free to move randomly and organize themselves arbitrarily; thus, the networks wireless topology may change rapidly and unpredictably. Mobile ad-hoc networks are infrastructureless networks since they do not require any fixed infrastructure, such as a base station, for their operation. The absence of fixed infrastructure make ad-hoc networks different from other wireless LANs. The traditional cellular networks are still, somehow, limited by their need of infrastructure (e.g., base station, router). For MANET this restriction is eliminated. In such a context, each node can act as a router, source and destination, and forward packets to the next node enabling them to reach their final destination via multiple hops. As shown in Figure 3.1, each node will be able to communicate directly with other nodes that reside within its transmission range. For communicating with nodes that reside beyond this range, the node needs to use intermediate nodes to relay messages hop by hop [2].

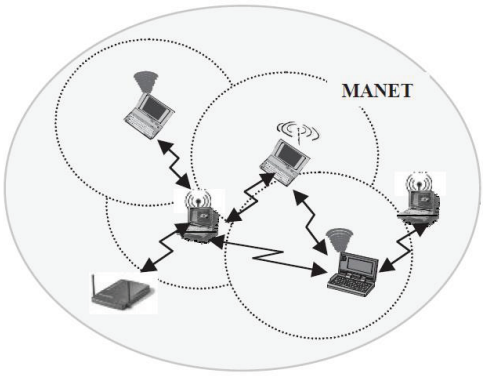


Fig. 3.1. Mobile Ad-hoc Network

MANETs have many benefits, such as self-reconfiguration, ease of deployment, and so on. However, this flexibility and convenience come at a price. Ad-hoc wireless networks inherit the traditional problems of wireless communications, such as bandwidth optimization, power control, and transmission quality enhancement, while, in addition, their mobility, multihop nature, and the lack of fixed infrastructure create a number of complexities and design constraints that are new to mobile ad-hoc networks such as publication, discovery and maintenance of configuration, as well as the addressing problem and the self-routing problem. MANET topology is highly dynamic and random. In addition, the distribution of the nodes and their capacity of self-organization

play an important role.
The main characteristics of the MANET can be summarized as follows:

- The topology is highly dynamic and frequent changes can be difficult to predict.
- They are based on wireless links, which have a lower capacity than their wired counterparts.
- Physical security is limited due to the wireless transmission.
- They are affected by higher packet loss rates and may have delays and jitter higher than in fixed networks due to the wireless transmission.
- They rely on batteries that can deplete their energy. As a result, the energy saving is an important design criterion of the system. In addition, the nodes must be sensitive to consumption: the set of features offered by a node depends on its available power (CPU, memory, etc..).

Since the nodes of a MANET are mobile, the routing management and energy consumption are critical issues. Wireless communication implies dependence on portable power sources such as batteries. However, due to the slow progress in battery technology, the battery continues to be a limited resource and thus saving energy in wireless networks is a key issue [110].

3.1.2 Energy Efficiency

Energy efficiency is the efficient use of energy and aims to reduce the amount of energy required to provide products and services. Reducing energy consumption has become one of the main goals in systems design. Low power consumption extends the operating hours of battery-powered systems, while a high energy consumption generates high temperatures and deteriorates the reliability. In addition, the growing concern about the environmental impact of electronic systems further highlights the importance of reducing power dissipation. The progress of hardware and wireless technologies has initiated a new era in which wireless devices provide continuous access to information regardless of their physical location, and actively participate in the creation of intelligent environments. As mobile devices are dependent on the battery power, it is important to minimize the energy consumption.
Here are listed the factors that affect the battery life of mobile phones and PDAs:

- Display.
- Vibration and alert tones.
- Play music.
- Bluetooth. Keep active Bluetooth causes a consumption of more than 20% in most of the battery. It is therefore necessary to keep the Bluetooth turned off when not in use, and it would be appropriate to use the least possible applications that use the Bluetooth to prolong the battery life.
- WLAN. The use of the free Wi-Fi and the ongoing search for access points affecting the consumption of battery.

- GSM network. The continuous search for 3G signal increases the demand on battery power, so it is recommended to disable any UMTS mode or dual.
- GPRS network. It is useful to turn off the packet data connection (by starting on request) instead of allowing the automatic search.
- Applications. The applications, after being used, must be completed and not left running in the background. The latter case, in addition to reducing the RAM memory available and, therefore, slow down the phone, it also causes a greater consumption of battery.

In particular, it can be significant the energy consumption of the network interface and applications with high computational load, respectively, the energy dissipated in the communication and the energy dissipated in the computation. In Mobile Computing, many operations are communication-dependent and the energy consumed by wireless communication represents more than half of the total power of the system [16].
Much of the research related to energy efficiency strategies has focused on ad-hoc wireless networks, due to their low implementation cost and low power consumption. Energy savings in wireless networks is a key issue. Well-designed architecture for MANETs involves all architectural layers, ranging from the physical to the application layer. Several methods have been proposed to reduce energy consumption. Each method can be classified in a particular architectural layer.

Physical layer. At the hardware level the aim is to reduce the energy consumed by a device using electronic components with lower energy consumption. For example, the CPU consumes energy and dissipates this energy, for switch devices contained in it (such as transistors) and for the impedance of the electronic circuits. Hence, it is important to design CPUs that execute these tasks efficiently without overheating. For mobile devices have been implemented particular CPU that consume very little, often, only a few hundred milliwatts of electrical energy. In comparison, the CPU of general purpose computer, such as desktop and laptop computers, dissipate much more energy due to their increased complexity and speed. These CPUs can consume hundreds of watts.
A way to reduce energy consumption, is to use smarter antennas or better batteries. However, the battery continues to be a limited resource, due to the slow progress in battery technology.
To control the energy consumption of the hardware (wireless network card, hard disk, CPU) is used a mechanism called *Power Management*. The latter is based on the use of software that turns off the power or switches the system to a low power state when idle. This is because, in many systems, some hardware components are not always used, so it is possible to adjust the power according to the computational load. This technique reduces the power consumption according to the demands generated by the running task. We can identify two states in which the device can be: the stand-by mode, in which

the device is inactive; and the working mode in which the device processes a task. In the first case the device is put into a state in which requires little power, while in the second case the device is put in a high power state.
Power management techniques are different. Two important techniques are the *Dynamic Voltage Scaling* and *Dynamic Frequency Scaling*, whereby the voltage of the CPU cores and the clock frequency can be altered to reduce the energy consumption at the cost of potentially lower performance. Another technique is the *Power Gating*, which allows processors to selectively shut down the internal circuitry. These techniques are applied according to a fixed policy for a specific device. Alternatively, they may also be applied at runtime, we talk in this case of *Dynamic Power Management*.

Data link layer. At this level one can act by using more efficient protocols that allow to avoid collisions and to reduce unnecessary retransmissions.

Network layer. As mentioned earlier, ad-hoc networks have a multi-hop nature, which means that all the nodes cooperatively maintain the connection without fixed infrastructure. Therefore, based on the topology of the network, a routing protocol must adapt its decisions to allow continuous communication between nodes. Each node is responsible for forwarding the packets to neighboring nodes, so it is necessary to balance the battery level of all the nodes, so as to avoid that the overloaded nodes run out of energy, disrupting the whole network. Different types of routing protocols have been proposed in the literature. In [17] has been measured and compared the energy consumption of four routing protocols, respectively Ad-hoc On Demand Distance Vector (AODV) [18], the Direct Source Routing (DSR) [19], the temporally-Ordered Routing Algorithm (TORA) [20] and Destination-Sequenced Distance-Vector Routing (DSDV) [21]. Firstly, authors in [17] select the most significant parameters of a MANET, at a later stage define and simulate a basic scenario, finally, by varying the selected parameters, generate and evaluate a wide range of scenarios quite different. The selected parameters were: the number of mobile nodes; the size of the area where the nodes move; the mobility model of the node; the number of traffic sources; and data traffic model. The results obtained from the simulations show that generally a pure on-demand protocol, such as DSR and AODV, performs better than the protocols DSDV and TORA. For all scenarios explored, TORA has the worst performance index. Moreover, increasing the number of nodes and leaving unchanged the number of traffic sources, TORA is not scalable, and energy consumption increases by 518% when nodes go from 25 to 50. DSDV has a behavior quite constant in all tested scenarios. The DSR normally performs better than AODV except in static networks in which the two protocols show a similar behavior. Comparing AODV and DSDV, there are several scenarios in which AODV has the worst performance of DSDV, usually when are allowed longer routes.

Transport layer. To reduce the energy consumption it can be used a more intelligent implementation of the transport layer, increasing reliability through techniques that operate flow control and congestion control.

Application layer. With the proliferation of portable computing platforms and small wireless devices, wireless ad-hoc networks have received increasing attention as a tool for data communication between devices. It is possible to manage the power consumption at high level, distributing the computational load among the nodes, so as to balance the load and maximize the life of the entire network. One approach is to define a middleware for unified management of the network. This thesis is based on this idea, which aims at defining a high-level scheduler that acts as an intermediary between the network nodes and the local schedulers. The implementation of these software modules requires the presence of distributed platforms, and it needs to find a fair trade-off between the energy consumed for communication and the energy consumed for computation.

3.1.3 Scheduling Algorithms

Scheduling is a key function of the operating systems that concerns almost all the resources of a computer. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating-system design. To describe a scheduling scenario is necessary to define a scheduling model, which includes:

- *Resources model.* It describes features, performance, availability and cost of resources.
- *Tasks model.* It describes the characteristics of the tasks.
- *Performance metrics.* Metrics used to measure performance which should be optimized.
- *Scheduling policy.* It includes the scheduling process that is a sequence of actions to be taken at relevant events, and the scheduling algorithm which defines how the task should be assigned to resources.
- *Programming model.* It is supplied to components/environments to interact with the scheduler describing the characteristics of an application.

A generic scheduling problem can be formally defined as shown below.

Tasks model $\mathcal{T}$

- *Tasks*: $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ is the set of n tasks which must be performed.
- *Interaction between tasks*: Data is a matrix $n \times n$, where Data(i, j) is the amount of data units (eg., byte) sent from t_i to t_j.
- *Precedence relationships between tasks*: it is defined a partial order on $\mathcal{T}$using the binary operator $<$: $t_i < t_j$ indicates that it is not allowed to run t_j before the execution of t_i is finished and Data(i, j) unit of data has been received from the processor on which t_j should run.
- *Execution time of each task on each processor*: ExecT is a matrix $n \times m$, where m is the number of processors and ExecT(i, j) represents the execution time of the task t_i when it is running on the processor p_j.

Generally, to represent the precedence relationships and the interaction between the tasks is used a graph, named tasks graph. A *tasks graph* is a directed acyclic graph (DAG), in which nodes represent the tasks and directed arcs are the precedence relationships. Each arc between two nodes has a weight that indicates the amount of interaction between the corresponding tasks.

Processors model $\mathcal{P}$

- *Processors*: $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$ is the set of m processors in the system.
- *Estimation of the communication delay between two processors*: CommT is a matrix, where CommT(i, j) represents the time required to deliver a unit of data from p_i to p_j.

Cost function $\mathcal{C}$

$\mathcal{C}$ is a cost function used to measure the goodness of a schedule. Two common cost functions are:

- *makespan*, represents the time in which all tasks have completed their execution;
- *comunicationTime + executionTime*, where comunicationTime is the total time spent during the interaction between tasks allocated to different processors, and executionTime is the total time spent for the execution of all tasks.

A generic scheduling problem can be formulated as follows:
Given a tasks model $\mathcal{T}$ and a processors model $\mathcal{P}$, look for a schedule $\mathcal{S}$ that map each task to a processor and determines the start time of each task so as to minimize a given cost function $\mathcal{C}$ such that:

- *overlap does not occur in the execution of tasks running on the same processor;*
- *all the precedence relationship, if any, are met.*

Classic scheduling algorithms

Different CPU-scheduling algorithms have different properties, and the choice of a particular algorithm may favor one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. Many criteria have been suggested for comparing CPU-scheduling algorithms. Which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best. The criteria include the following:

- *CPU utilization.* A measure is the percentage of CPU use. The CPU must be as busy as possible.
- *Throughput.* CPU is busy when performing the work. One measure of work is the number of processes that are completed per time unit, called throughput.

- *Waiting time*. A measure is the amount of time that a process spends waiting in the ready queue.
- *Response time*. In an interactive system, turnaround time may not be the best criterion. Often, a process can produce some output fairly early and can continue computing new results while previous results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced.

It is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time. In most cases, we optimize the average measure. However, under some circumstances, we prefer to optimize the minimum or maximum values rather than the average [10]. There are many different CPU-scheduling algorithms. In this section, we describe several of them.

First-Come, First-Served Scheduling

By far the simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. Within this scheme, the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue. FCFS scheduling algorithm is nonpreemptive. The average waiting time under the FCFS policy is often quite long, and may vary substantially if the processes CPU burst times vary greatly. A problem with the FCFS algorithm is the convoy effect, that happens when all the other processes wait for the one big process to get off the CPU. This effect results in lower CPU and device utilization than might be possible if the shorter processes were allowed to go first.

Shortest-Job-First Scheduling

The shortest-job-first (SJF) scheduling algorithm associates with each process the length of the processs next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie. The SJF algorithm can be either preemptive or nonpreemptive. The next CPU burst is generally predicted as an exponential average of the measured lengths of previous CPU bursts.

Priority Scheduling

A priority is associated with each process, and the CPU is allocated to the process with the highest priority. Equal-priority processes are scheduled in FCFS order. A priority scheduling algorithm can be either preemptive or nonpreemptive. A major problem with priority scheduling algorithms is indefinite blocking, or starvation. A process that is ready to run but waiting for the CPU can be considered blocked. A priority scheduling algorithm can leave some low priority processes waiting indefinitely. A solution to the problem of

indefinite blockage of low-priority processes is aging. Aging involves gradually increasing the priority of processes that wait in the system for a long time.

Round-Robin Scheduling

The round-robin (RR) scheduling algorithm is designed especially for time-sharing systems. It is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes. A small unit of time, called a time quantum or time slice, is defined. A time quantum is generally from 10 to 100 milliseconds in length. The ready queue is treated as a circular queue. The performance of the RR algorithm depends heavily on the size of the time quantum. At one extreme, if the time quantum is extremely large, the RR policy is the same as the FCFS policy. In contrast, if the time quantum is extremely small (say, 1 millisecond), the RR approach can result in a large number of context switches. Thus, we want the time quantum to be large with respect to the contextswitch time. A rule of thumb is that 80 percent of the CPU bursts should be shorter than the time quantum.

Multilevel Queue Scheduling

Another class of scheduling algorithms has been created for scenarios in which processes are easily classified into different groups. For example, a common division is made between foreground (interactive) processes and background (batch) processes. These two types of processes have different response-time requirements and so may have different scheduling needs. A multilevel queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, and each queue has its own scheduling algorithm. In addition, there must be scheduling among the queues, which is commonly implemented as fixed-priority preemptive scheduling.

Multilevel Feedback Queue Scheduling

The multilevel feedback queue scheduling algorithm is a multilevel queue scheduling that allows a process to move between queues. The idea is to separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it will be moved to a lower-priority queue. This scheme leaves I/O-bound and interactive processes in the higher-priority queues. In addition, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.

Our discussion thus far has focused on the problems of scheduling the CPU in a system with a single processor. If multiple CPUs are available, load sharing becomes possible, but scheduling problems become correspondingly more complex [10]. We focus on systems in which processors are homogeneous in terms of their functionality.

One approach to CPU scheduling in a multiprocessor system has all scheduling decisions, I/O processing, and other system activities handled by

a single processorthe master server. The other processors execute only user code. This *asymmetric multiprocessing* is simple because only one processor accesses the system data structures, reducing the need for data sharing.

A second approach uses *symmetric multiprocessing* (*SMP*), where each processor is self-scheduling. All processes may be in a common ready queue, or each processor may have its own private queue of ready processes. In both cases, scheduling proceeds by having the scheduler for each processor examine the ready queue and select a process to execute. Because of could be multiple processors trying to access and update a common data structure, the scheduler must be programmed carefully. We must ensure that two separate processors do not choose to schedule the same process and that processes are not lost from the queue. Virtually all modern operating systems support SMP, including Windows, Linux, and Mac OS X.

Energy-aware scheduling algorithms

The traditional scheduling algorithms do not take into account the energy consumption, on the contrary, they focus on the performance and execution times of applications. Instead, an energy-aware scheduling has the objective of minimizing the energy consumption both in terms of energy consumed for the execution of applications and for communication.
An emerging theme in the field of resource management is energy saving in distributed systems. Several papers proposed energy management for server computers that are focused on energy optimization techniques in multiprocessor environments, such as [111, 112]. Another proposal for balancing the energy load and performance optimization in this type of environments can be found in [113]. Other approaches are also used to manage the resources of a shared server, for example in [114], the authors use greedy allocation of resources by distributing the workload among the different web servers assigned to each service. This technique reduces the energy use of servers by 29% or more for a typical Web workload. The work presented in [22] proposes a hybrid architecture that integrates low-power systems and high-performance ones.
In [23] is proposed a formulation of mixed integer linear programming (MIP) to dynamically configure the consolidation of multiple services/applications in a cluster of virtualized servers. The approach is centered on energy efficiency and takes into account the cost of activation/deactivation of servers. It also takes into account the web workload and the decision algorithm is run every five minutes so as to be able to manage heterogeneous workloads and adapt the system to each new task comes in, making the most of the energy. However, the use of heavy mathematical calculations in the programming can lead to a too slow decision-making process for an online planning.
In [24] it is presented an approach that uses virtualization for consolidation and an approach of dynamic configuration to optimize the energy consumption in clusters of virtualized servers, proposing an algorithm to dynamically manage those clusters. Following the same idea, the work presented in [25]

aims to reduce the energy consumption of virtualized data centers by encouraging the migration of VMs and optimizing the placement of VMs, reducing human involvement.
To reduce the energy consumption in clusters, other approaches used are those of *machine learning*. In [26] a reinforced learning approach for the simultaneous and on-line management of performance and power consumption is presented. These policies allow to save more than 10% of servers energy, keeping the performance close to a desired target.
Das et al. [27] present an approach that uses multi-agent technology in order to shut down the server in low-load conditions, achieving an energy saving of 25% without incurring in SLA penalties. All of these approaches use a reinforced learning in order to know the management policies of the data provided.
Filani et al. [115] offer a solution that includes a platform whit a Policy Manager, which controls the energy and temperature sensors, and enforces policies for energy management. It is also shown how the power management can be used as a basis for energy management in data centers.
The approach proposed in [28] uses two different mechanisms in order to reduce the energy consumption of a data center with respect for different SLAs. One of the mechanisms that allows to save energy is to turn off the machines idle, which saves more than 200W on the machines tested.
In [29] different scheduling policies are applied to assign jobs to available machines and redistribute the jobs in execution, in order to make some machines idle and then turn them off. Note that, reactivate the machines when they are necessary, it is not an immediate process and without costs, and this overhead, which can take more than a minute, must be taken into account. They used different traditional scheduling policies, including: *Random* that assigns tasks in a random way; *Round Robin* that handles the tail of the resources as a circular queue by assigning a task to the first available node, this implies a maximization of the amount of resources assigned to a task, but also a poor use of resources; *Backfilling* that tries to fill as much as possible nodes, thus solving the above problem; *Dynamic Backfilling* that is able to move (i.e., migrate) tasks between nodes in order to provide a higher level of consolidation taking into account parameters such as the use of the system, the performance of the current job, or the satisfaction of various expected SLA. While the Dynamic Backfilling performs well when you have precise information, other policies are necessary when the information is incomplete or inaccurate.
Energy-aware scheduling algorithms for single-processor systems have been developed. These are based on *Power Management* techniques that aim to make efficient use of the underlying hardware, switching the system to a low power state when inactive. Two energy-aware scheduling algorithms for single-processor systems have been proposed in [30, 31]. The algorithm proposed in [30], orders the execution of the task so that idle periods are grouped together rather than scattered. Turning off the device when it is idle, and reactivate it when it has to serve requests, introduces a consumption of time and energy. Therefore, the algorithm turns off the devices only when the downtime

are continuous and long enough. The algorithm described in [31] introduces a new metric to measure the energy performance of the CPU, *millions-of-instructions-per-joule* (*MIPJ*). It adapts the speed of the CPU and the voltage level so as to increase the number of instructions executed per unit of energy.
The energy-aware algorithms for multiprocessor systems have not gotten enough attention. The most significant contribution to this issue comes from two research groups, Jah et al. ([36, 37, 38]) and Hu and Marculescu ([32, 33, 34]). In [32], has been formulated an energy-aware scheduling problem for some architectures Network-on-Chip (NoC) for embedded systems, and has been defined a heuristic to solve it. The algorithm aims to meet the deadlines of tasks with minimal energy consumption. For the evaluation of performance, each application running in the system is described as a graph, called Communication Task Graph (CTG), moreover, they adopt a metric to measure the energy dissipation in the network, originally proposed in [33]. Luo and Jah [36] propose an iterative energy profile guided by an allocation algorithm that dynamically scale frequency and voltage of processors, *Dynamic Voltage Frequency Scaling* (*DVFS*). In [37] the authors apply the technique of simultaneously scaling voltage processors and communication links. The work in [38] focuses again on voltage scaling of processing elements and combines the DVFS with adaptive body biasing (ABB), a technique that has its advantages, especially in the field of technology with dimensions smaller than 50nm and with clock frequencies of several GHz.
Topcuoglu et al. [35] present two scheduling algorithms with low complexity and effective performance in heterogeneous environments, *Heterogeneous Earliest- Finish-Time* (*HEFT*) and *Critical Path-on-a-Processor* (*CPOP*). In addition, their work provides a good overview of different scheduling heuristics. Thormann [123] analyzes the energy consumption of different allocation algorithms. His work, however, assumes independent tasks that do not have deadlines. Therefore its results can not be applied in contexts in which it is required to comply with precedence constraints and time constraints.

Scheduling on mobile devices

Several scheduling algorithms for mobile devices have been presented and implemented, even in sensor networks. Below is a brief overview of these not-energy-aware algorithms.

Vehicle Routing Problem (VRP)

This is a logistic problem whose objective is to minimize the total distance traveled by vehicles while they serve all customers. Solve the problem using optimization models is computationally prohibitive for real-world applications, it is NP-hard. Therefore, several heuristic methods have been applied to the above problem. The most basic VRP is the *Capacitated Vehicle Routing Problem* (*CVRP*), which assumes a fixed set of vehicles of uniform capacity in a

central depot. The *Vehicle Routing Problem with Time Windows* (*VRPTW*) is a generalization of the CVRP with the added complexity of time windows and other relevant temporal data. The time window can be both hard and soft. In the case of hard window, a vehicle can arrive earlier and wait for the window to open, but he is not allowed to arrive late. In the case of soft window, a vehicle will be allowed to arrive late at the expense of cost. This problem can be extended to multiple devices using the *Multi-depot Heterogeneous problem*. Since the problem extends to heterogeneous vehicles, for a feasible solution must satisfy the following constraints [116]: (i) each route must start and end at the same depot, (ii) each node must be served by only a single vehicle; (iii) the total load assigned to the vehicle shall not exceed its carrying capacity, (iv) the service time length must not exceed the maximum working time allowed; (v) the collection/delivery to each site should be within the time window. The aim is to minimize the total performance cost of services of collection/delivery of all nodes.

Heuristic algorithms

Some of the heuristic solutions for scheduling problem on mobile devices are *Earliest Deadline First* (*EDF*), *EDF with k-Lookahead* and *Minimum weighted Sum First* (*MWSF*) [40]. In EDF, first the node with the nearest deadline is visited. The algorithm is based only on the deadline, and is not taken into account any other factor. To improve this approach, it has been proposed EDF with lookahead where first the node with the nearest deadline is visited, and this node is chosen by permuting the k nodes with the nearest deadline. The permutation is chosen such that none of the k nodes lose its deadline. For each visit, k nodes are permuted and the next node to visit is chosen on the basis of the values permuted. The third algorithm is MWSF, that in determining how to schedule visits takes into account the weight of the deadlines as well as the distance between the nodes. Although the performance of the algorithm MWSF are considered the best, the occurrences of the round-trip movements between distant nodes are common.

Partitioning-Based Scheduling (PBS) Algorithm

In the PBS [41] algorithm all nodes are partitioned into different groups, called containers, in such a way that the nodes in the same container have similar deadlines and are close to each other. Within a single container, the well-known *Traveling Salesman Problem* (*TSP*) is solved by calculating the tour with minimal cost that allows to visit each node exactly once. In the end, the schedule of each group are concatenated to form a complete schedule. A numerical computation is performed to determine the lower bound on the minimum speed of the moving parts, getting an an approximated value. In the EDF with k-lookahead algorithms and MWSF, the constraints of cost and speed are not considered and these bring changes in the resulting values.

Energy-aware Scheduling on mobile devices

The energy-aware computing has become popular in both large systems consisting of multiple processing units, to reduce energy consumption and cooling costs, both in systems of mobile computing to extend battery life. Since processors consume a large percentage of power in computer systems, especially in embedded systems, much research has been conducted on the management of energy consumption of processors. The traditional scheduling algorithms, based on criteria that aim to improve performance and decrease the execution time, are not suitable for systems in which the most valuable resource is the energy of the battery. Therefore, have been proposed scheduling algorithms whose goal is to minimize the power consumption so as to prolong the battery life of mobile devices.
Most of the existing research work in the area of energy-aware systems concerns hardware-based techniques focused on reducing the energy consumption of the processor. Among those techniques, turning off idle components is widely adopted [44]. Dynamic Voltage Scaling (DVS) is another hardware-based technique for energy conservation, allowing for simultaneously varying the processor voltage and frequency as per the energy performance level required by the tasks [45, 48, 47]. Accordingly, several energy saving scheduling (e.g. [46, 39]) algorithms for multi-processor distributed systems are focused on architectures with dynamic voltage scaling or dynamic power management capabilities. Among software-based energy-aware techniques, *remote execution* provides that a device with limited energy transfers a computational task to a nearby device which is more energy powerful [2, 110]. This approach was made feasible by recent developments in wireless technologies. A wireless system, in which devices cooperate among themselves, reduces the effect of this limitation through the distribution of the computational load among all devices of the system so as to achieve energy efficiency. This approach has led to the need for *remote execution platforms* that allow the migration of computational tasks to a remote server, and predict the resource usage of each task on the local machine and on a remote one. In [43] it is proposed a platform for remote execution mechanism based on the client-server called *Remote Processing Framework* (*RPF*). On the client side, RPF maintains a database where it stores the statistics of resource usage by the various task and decide whether to use local or remote execution. On the server side, when it is preferable to remote execution, RPF migrate the task and returns the results to the client. The decision-making process in RPF is simple and is obtained by comparing the performances remote and local, and choosing the best among the two.
In [15], it is proposed *Spectra*, a component of remote execution of an operating system designed for mobile and pervasive devices. Spectra monitors various resources and predicts the use of these resources by the task. To monitor the batteries and predict the energy consumption of tasks, leverages the advent of smart batteries that provide information on the level of energy and

on energy consumption. Spectra estimates the local execution and a set of remote executions and chooses the best of these.
Another platform for remote execution, primarily for PDAs, is the *Remote Processing Platform* (RPP) [42]. This platform makes use of smart batteries and the complexity of computational tasks to predict the energy consumption of the tasks, and keeps a log database that stores information about the different computational tasks. The decision-making process in the RPP is similar to that of Spectra and RPF.
Energy-aware task scheduling is another software method where the scheduling policy aims at optimizing the energy consumption. In the work presented in this thesis we adopt this method. To the best of our knowledge, little work has been done on energy-aware scheduling over a MANET. Alsalih et al. [49] proposed an energy-aware dynamic task allocation algorithm over MANETs. However, this work is different from ours in terms of the underlying architecture and cost function to be optimized. We clustered the devices to promote local cooperation among nearby devices and minimize the transmission energy. This issue is particularly relevant because we have experimentally found that the transmission energy highly impacts on the overall energy consumption. In contrast to ours, the solution proposed in [49] does not address the communication aspects of the system. Furthermore, we adopt a different objective function: we maximize the network lifetime rather than minimizing the energy consumption. Using the network lifetime parameter we are able to actually consider the real energy consumption rate of single devices, single clusters and the overall network. Conversely, [49] does consider only the local computation issues and works at a node level without taking into account the workload in the rest of the network.

3.2 Mobile Architecture and Energy Models

This section presents the energy-aware architecture of the system and it describes the software structure of mobile devices. At a later stage illustrates the energy model adopted to minimize energy consumption and maximize the life time of the whole network of mobile devices.

3.2.1 M2M Architecture

The architecture of the system, referred to as *Mobile-to-Mobile* (*M2M*) architecture, is designed to enable the execution of mobile-to-mobile applications having energy efficiency as the primary goal. We consider a multi-hop wireless ad-hoc mobile network architecture, depicted in Figure 3.2, that allow on-demand collaborations among mobile nodes.
In a MANET that changes its topology dynamically, efficient resource allocation, energy management and routing can be achieved through adaptive clustering of the mobile nodes. In order to promote and easy collaborations

when two or more mobile users meet each other, we let them grouping into clusters referred to as mobile groups. In our clustering scheme the mobile nodes are divided into virtual groups. Specifically, geographically adjacent devices are assigned to the same cluster [57]. Consequently, the proposed architecture includes a number of *mobile groups* or *clusters*. Under a cluster structure, mobile nodes may be assigned a different function, such as cluster-head or cluster member. A *cluster-head* serves as the local coordinator for its cluster, performing intra-cluster transmission arrangement, data forwarding, and so on. A *cluster member* is usually called an ordinary node, which is a non cluster-head node without any inter-cluster links. Figure 3.2 shows the interactions among the different components of the architecture. Mobile nodes within a group interact trough ad-hoc connections (e.g., wi-fi, bluetooth) that we refer to as M2M links. Interactions among mobile groups (cluster-to-cluster links) take place through ad-hoc connections among the cluster-heads of the groups. All types of interactions take place either to ask for a computation request or to cooperate in order to collaboratively execute a computational task.

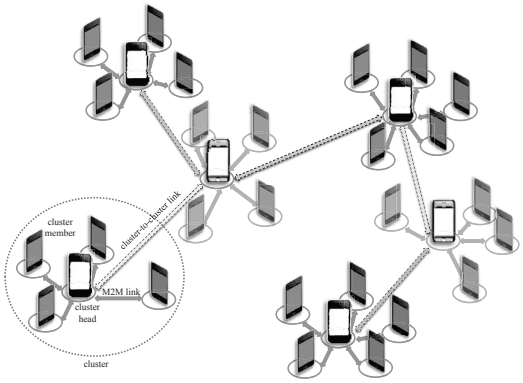


Fig. 3.2. Mobile-to-Mobile architecture

Mobile Device Components

Mobile nodes include a set of software components that cooperatively perform a group of functionalities. As shown in Figure 3.3, each node includes four software components:

1. Resource Information Service (RIS);
2. M2M Coordination Service (MCS);
3. Energy-aware Scheduler (EAS);
4. Application Service (APPS).

The *RIS* is responsible to collect information about all the resources inside a mobile node and the context in which an application is running in order to adapt its execution. To this aim, the RIS is composed of two modules implementing the above cited features:

- *Resource Monitoring module.* It informs the system about the mobile device resources measurement such as the available memory, CPU utilization, battery consumption, battery level, remaining time to fill memory, network connectivity performance.
- *Resource Evaluator module.* This module acts as a resource measurement receiver from both local and environmental resources. It then takes some actions on the basis of the received measures, i.e. choosing the most suitable configuration for the task. Moreover, the module is responsible for starting the task with the appropriate parameters.

The *MCS* is responsible for the coordination among mobile devices and includes two modules:

- *Mobile-to-Mobile Management module.* It includes mechanisms aiming at the coordination of the nodes within a group such as cluster formation and maintenance, joining of a new node to a group, cluster-head election, cooperative task execution.
- *Cluster-to-Cluster Management module.* This module provides mechanisms allowing mobile devices to organize themselves into clusters, cluster-to-cluster interactions to the end of a task allocation, coordination to collaboratively execute a task.

The *EAS* is the component responsible for task assignment among local groups. It implements a scheduling strategy aimed at prolonging network life-time by distributing energy consumption among local groups. In such an approach, whenever a resource limited computing device (client) has a set of tasks to be executed, it uses the energy resources in nearby computing devices (servers) and an efficient task assignment is found in such a way that the total consumed energy is minimized. The scheduler interacts with the RIS through its resource-monitoring and resource-evaluator modules. Moreover, the scheduler is also tightly related to the APPS component, as it is actually the scheduler that activates a process. The EAS includes three modules:

- *Cost Estimator module.* This module exploits information about availability, performance and cost of resources collected by the RIS component. It deals with the actual calculation of the estimation functions on the basis of the perceived status of resources w.r.t. energy and memory constraints.
- *Mapper module.* This module schedules the tasks. It embeds a scheduling algorithm, and a matchmaker that takes into account resource characteristics, incorporates interdependencies among resource groups or types, and computational and I/O cost evaluations to map the available resource units to newly scheduled tasks according to a pre-specified mapping objective function.

- *Scheduling Process module.* This module guides the scheduling activity. It receives jobs, requests the corresponding schedules to the mapper, and orders the execution of scheduled tasks.

The APPS is responsible for the execution of a task over a mobile device. It includes two modules:

- Data Collection module. This module provides access/store mechanisms for data to be processed or generated as a result of a process. Typically, only a limited amount of data can be stored on a mobile device. Therefore, this module will manage the interaction with a stationary node that will act as a storage node or as a source for data.
- Running Task module. It is responsible for managing the execution of a task on the mobile device.

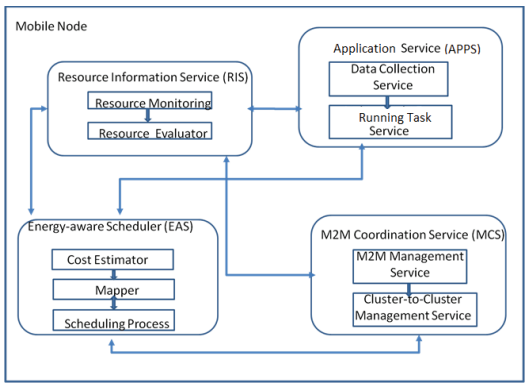


Fig. 3.3. Software components of a mobile device.

3.2.2 Energy Model

Energy consumption of mobile devices depends on the computation and the communication loads. We define E_i as the energy consumption of device d_i in a time interval δt, which is the sum of energy consumption for communication, ET_i, and computation, EC_i, of all the tasks assigned to device d_i within time interval δt:

$$E_i = EC_i + ET_i \tag{3.1}$$

We estimate the energy consumption for computation and analytically evaluate the energy consumed for communication. This last issue is the main aim of the section.

Communication Energy Model

In MANET networks nodes must always be ready to receive traffic from neighbors due to the absence of base station nodes. Thus, a network interface operating in ad-hoc mode has to continuously listen to the wireless channel, consuming this way a constant idle energy power. Therefore, each node *overhears* every packet transmission occurring in its transmission range consuming this way energy uselessly. This idle energy consumption is referred to as *overhearing*. Due to overhearing, a new cost in the computation of per-packet energy consumption is introduced and it is the cost for discarding overheard packets. Therefore, to model the energy consumed for communication, the costs to send, receive and discard a packet must be included. Consequently, the energy consumed by a device d_i for communication can be defined by the following equation:

$$ET_i = E_{send_i} + E_{receive_i} + E_{discard_i} \tag{3.2}$$

A packet may be sent through a broadcast or a point-to-point channel. With the former the packet is received by all hosts within the sender's transmission range; with the latter the packet is discarded by non-destination hosts.

We assume a commonly used wireless propagation model where the received signal power attenuates proportionally to $TR^{-\alpha}$, where TR is the transmission range and α is the loss constant, typically between 2 and 4 depending on the wireless medium [4]. Moreover, according to [117] we evaluate the energy consumption behavior of the mobile ad hoc network based on a model where the cost for a node to send or receive a message is modeled as a linear function. In this function there is a fixed cost associated with channel acquisition and an incremental cost proportional to the size of the message. The fixed channel access costs, denoted as b_{send} and b_{recv}, and the incremental costs, m_{send} and m_{recv}, are the same for broadcast and point-to-point. For point-to-point traffic, the fixed cost includes also the MAC negotiation. In the IEEE 802.11 MAC protocol, the source sends an *RTS* (request-to-send) control message, identifying the destination. The destination responds with a *CTS* (clear-to-send) message. Upon receiving the *CTS*, the source sends the data and awaits an ack from the destination. For simplicity, these small control messages are assumed to have the same fixed send ($b_{sendctl}$) and receive ($b_{recvctl}$) costs.

The cost, $E_{send_{ij}}$, for a node d_i to send a point-to-point packet to a node d_j is described by the following equation:

$$E_{send_{ij}} = b_{sendctl} + b_{recvctl} + m_{send} * |MSG| + b_{send} + b_{recvctl} \tag{3.3}$$

where $|MSG|$ is the size (number of bits) of the message exchanged among nodes d_i and d_j.

The cost to receive a point-to-point packet is modelled through the following equation:

$$\mathrm{E}_{\mathrm{rec}_{\mathrm{ij}}} = \mathrm{b}_{\mathrm{recvctl}} + \mathrm{b}_{\mathrm{sendctl}} + \mathrm{m}_{\mathrm{recv}} * |\mathrm{MSG}| + \mathrm{b}_{\mathrm{recv}} + \mathrm{b}_{\mathrm{sendctl}} \tag{3.4}$$

In case of broadcast transmission, the cost to send a packet is represented through Equation 3.5 while the cost to receive a packet is given by equation 3.6:

$$\mathrm{E}_{\mathrm{send}_{\mathrm{broad}_{\mathrm{i}}}} = \mathrm{m}_{\mathrm{send}} * |\mathrm{MSG}| + \mathrm{b}_{\mathrm{send}} \tag{3.5}$$

$$\mathrm{E}_{\mathrm{rec}_{\mathrm{broad}_{\mathrm{i}}}} = \mathrm{m}_{\mathrm{recv}} * |\mathrm{MSG}| + \mathrm{b}_{\mathrm{recv}} \tag{3.6}$$

Thus, the energy cost of node $\mathrm{d_i}$ for sending (Equation 3.7) and receiving (Equation 3.8) packets depends on the used transmission mode:

$$\mathrm{E}_{\mathrm{send}_{\mathrm{i}}} = \begin{cases} \mathrm{E}_{\mathrm{send}_{\mathrm{ij}}} \text{ if point-to-point} \\ \mathrm{E}_{\mathrm{send}_{\mathrm{broad}_{\mathrm{i}}}} \text{ otherwise} \end{cases} \tag{3.7}$$

$$\mathrm{E}_{\mathrm{receive}_{\mathrm{i}}} = \begin{cases} \mathrm{E}_{\mathrm{rec}_{\mathrm{ij}}} \text{ if point-to-point} \\ \mathrm{E}_{\mathrm{rec}_{\mathrm{broad}_{\mathrm{i}}}} \text{ otherwise} \end{cases} \tag{3.8}$$

Non-destination nodes within the transmission range of either the transmitting or receiving nodes overhear the traffic. The cost of discarding is comparable to the one of a broadcast receiving:

$$\mathrm{E}_{\mathrm{discard}_{\mathrm{i}}} = \mathrm{m}_{\mathrm{discard}} * |\mathrm{MSG}| + \mathrm{b}_{\mathrm{discard}} \tag{3.9}$$

In this thesis, we refer to the results of some experiments that estimate the energy consumed by a network interface based on the IEEE 802.11 protocol. The experiments are described in [50]. The detailed steps are presented as a set of linear equations. In these experiments has been used an oscilloscope to measure the voltage and current flow in input to the network interface in steady state, and have been sent and received packets of various sizes. To compute the energy cost coefficients per-package was used linear regression. The results are reported in Table 3.1.

Table 3.1. Communication energy cost coefficients of IEEE 802.11-based wireless network interface.

		$\frac{m \cdot w \cdot S}{byte}$	$byte$	$\frac{m \cdot w \cdot S}{byte}$
Point-to-Point send	$E_{send_{ij}}$	1.9	$\times size$	420
Broadcast send	E_{broad_i}	1.9	$\times size$	250
Point-to-Point and Broadcast receive	$E_{recv_{ji}}$	0.42	$\times size$	330

3.3 Energy Aware Scheduler

In this section we present the Enery-Aware (EA) scheduling strategy over the reference mobile-to-mobile network architecture introduced in Section 3.2.1.

As mentioned above, the main design principle of the EA strategy is finding a task allocation that prolongs the total lifetime of the network, by balancing the energy load among all the devices.

Since applications change behavior as battery drains, the system is highly dynamic and the topology change very fast according to the clustering scheme described above. Here, we explore how applications can dynamically modify their behavior to conserve energy in scheduling tasks over the network. To guide such adaptation, the operating system monitors energy supply and demand (see the RIS component in the architecture in Figure 3.3).

3.3.1 Scheduling Model

The scheduling problem is the process of mapping a given application onto a target architecture by: (1) selecting which task of the application shall be executed; (2) allocating that task to a resource; (3) computing start and execution times for the task; (4) repeating these steps until all tasks are scheduled. It is common in the literature to use the terms task allocation and task scheduling interchangeably. However, scheduling is commonly used to describe all of the above mentioned steps as well as to describe the computation of start and execution times only. Task allocation is, therefore, a step of the more general scheduling problem; it can also be seen as a global scheduling or meta-scheduling that distributes the tasks among the devices. Once tasks have been allocated, the problem becomes one of defining a feasible local schedule that manages task execution for each node. We refer to task allocation or task scheduling interchangeably.

We introduce the following model to support the description of the scheduling strategy.

- $\mathcal{D}= \{d_1, d_2, ..., d_m\}$ is the set of devices in the network.
- $\mathcal{T}= \{t_1, t_2, ..., t_n\}$ the set of tasks to be executed, which are assumed to be independent each other.

A generic task t_i is characterized by the following features:

- execution time of t_i to process a data set of size s on a device d_j;
- energy consumption of t_i to process a data set of size s on a device d_j;
- memory consumption of t_i to process a data set of size s on a device d_j;

Before going into the details of the scheduling policy, some notations are introduced.

- $M_i(t)$: memory availability of device d_i at time t.
- $RE_i(t)$: residual energy available at device d_i at time t.
  $P_i(t)$: instantaneous power of device d_i at time t.
- $EEC_i(t_j, s)$: estimated energy consumed for computation by device d_i to run a task t_j over a data set of size s.

- $EET_i(t_j, s)$: estimated energy consumed for communication by device d_i to run a task t_j over a data set of size s.
- $EMC_i(t_j, s)$: estimated memory consumption of device d_i to run a task t_j over a data set of size s.

Given this model, we define the concept of *residual life* $RL_i(t)$ of device d_i at time t, as follows:

$$RL_i(t) = RE_i(t)/P_i(t). \tag{3.10}$$

Similarly, we define *network lifetime* at time t, $RL_{net}(t)$, as the total residual life of the devices in the network. Therefore, the goal of our scheduling strategy is finding a task allocation that maps each task $t_i \in \mathcal{T}$ to a device $d_j \in \mathcal{D}$ so as to maximize $RL_{net}(t)$.

3.3.2 Scheduling Strategy

This section details the proposed energy-aware (EA) dynamic scheduling scheme. Our strategy is able to deal with a system and its fast changing topology where independent tasks arrive at each node dynamically over time.

The scheduling problem has been proven to be NP-Complete in its general form [51]. Even though optimal algorithms have been proposed for some restricted versions of the problem, heuristic-based algorithms have been proposed for the more general versions of the problem allowing to find good allocations in polynomial time [52].

Accordingly, we propose a two-phase heuristic-based decentralized algorithm. When an assignment decision has to be made for a task, the first phase, referred to as *local assignment*, tries to assign the task locally to the cluster where the execution request has been generated, so as to maximize the cluster residual life. If the local assignment failed because none of the devices within the cluster can execute the task, the cluster-head activates the second phase, referred to as *global assignment*, responsible for task arbitration among clusters. In this case, the task will be assigned to the most suitable device, all over the network of clusters, that maximizes the overall network lifetime.

We formalize the problem of task allocation as an optimization problem. As said before, the aim of the optimization is to maximally extend the network lifetime and the number of alive devices. To this aim, tasks are allocated to the devices such as to balance the load among them. We optimize the problem by iteratively trying to improve a candidate solution. A feasible allocation is optimal if the corresponding cluster residual life (in case of local assignment) or network lifetime (in case of global assignment) is maximized among all the feasible allocations.

A device d_i can be candidate for the assignment of a task t_a, if it satisfies the following constraints:

1. d_i has enough memory to perform t_a over a data set of size s, i.e.

$$EMC_i(t_a, s) < RE_i(t).$$

2. d_i has enough energy to perform t_a over a data set of size s, i.e.

$$EEC_i(t_a, s) < RE_i(t).$$

During the local assignment phase, a cluster-head, or the set of neighboring cluster-heads in case of the global assignment, will choose the local node, among the ones satisfying the above constraints, that will prolong the life of the corresponding local group by using the following objective function:

$$RL_{LG_j}(t) = Max \sum_{i=1}^{N_{LG_j}} \alpha_i RL_i(t) \tag{3.11}$$

where RL_{LG_j} denotes the residual life of local group LG_j, N_{LG_j} is the number of nodes within the local group LG_j, RL_i is the residual life of node d_i in the group, and parameter α_i takes into account the importance of d_i in the local group. The node associated with the maximum value of the objective function will be selected by the cluster-head as candidate node. Note that throughout the experimental evaluation the parameter α_i is set to 1 thus, all the nodes have the same role within the local group.

If the global assignment phase is activated, the final decision is taken by considering all the candidate nodes proposed by the neighboring clusters. The task will be assigned to the local group that maximizes the network lifetime:

$$RL_{net}(t) = Max \sum_{j=1}^{N} \alpha_j RL_{LG_j}(t) \tag{3.12}$$

where N is the number of clusters in the network.

3.3.3 Scheduling Algorithm

When a node, referred to as *requesting node*, wants to execute a task it requests its cluster-head to handle the task assignment process (see Algorithm 1).

```
Method allocateTaskReq
Input: task t, data set size s

begin
   if (this.isClusterHead()) then
      allocateTask(t, s);
   else
      myClusterHead.allocateTaskReq(t, s);
   end
end
```

Fig. 3.4. ALGORITHM 1: Task allocation request.

Upon receiving a task allocation request, the cluster-head triggers, through the method shown in Algorithm 2, the activation of the task allocation strategy to find the optimal assignment for that task. To this aim, the cluster-head

verifies whether the residual life of its group is greater than the 20% of its peak value. If the check is successful, it starts up the local assignment phase. If the check is negative or the local assignment failed because none of the nodes within the cluster can execute the task, the cluster-head activates the global assignment phase. Note that in case of negative check on the threshold of the local group residual life, the global phase is activated only if is found at least a neighboring group whose residual life is greater than a given threshold. If this last condition is not satisfied, the local phase is activated anyhow. The choice of a low threshold value (20% in our case) originates from the experimental observation that the transmission energy heavily impacts on the energy consumption of about one magnitude order compared to the computation energy [53]. This is also the reason for pursuing first a local optimum, because this allows reducing the transmission costs.

```
Method allocateTask
Input: task t, data set size s

begin
    localAssignmentFailed ← false;
    C ← determineNeighboringClusters();
    if (RL_LG_i > 20% or C=∅) then
        < d_candidate, ERL_LG_i > ← localAssignment(t, s);
        if (< d_candidate, ERL_LG_i >≠ ∅) then
            if (d_candidate = this) then
                startTask(t, s);
            else
                d_candidate.executeTask(t, s);
            end
        else
            localAssignmentFailed ← true;
        end
    end
    if ((RL_LG_i ≤ 20% or localAssigmentFailed) and C ≠ ∅) then
        globalAssignment(t, s, C);
    end
end
```

Fig. 3.5. ALGORITHM 2: Task allocation by the cluster-head of local group LG_i.

The scheduling starts with the local assignment phase where the cluster-head tries to assign the task to a node within the group that maximizes the cluster residual life (see Algorithm 3). In this way, the cluster-head determines a candidate node in the group meeting the requirements to execute the task in terms of energy and memory constraints. The local assignment phase can be executed both by the cluster-head of the requesting node and by the cluster-heads of neighboring clusters involved in a global assignment.

To limit the communication costs in the network, the global assignment phase (Algorithm 4) considers the neighboring clusters distant at most three hops from the requesting cluster-head. Among the neighboring clusters are selected only the ones having a residual life greater than a given threshold

```
Method localAssignment
Input: task t, data set size s
Output: candidate node d_candidate, estimated residual life
ERL_LG_i of local group LG_i if t is assigned to d_candidate

begin
   if (this = CH_i or (this ≠ CH_i and RL_LG_i > 30%)) then
      RL_LG_curr ← 0;
      ERL_LG_i ← 0;
      d_candidate ← ∅;
      foreach node d_i ∈ LG_i do
         if (d_i.hasSkill(t, s)) then
            RL_i ← d_i.estimateNodeResidualLife(t, s);
            RL_LG_curr ←estimateGroupResidualLife(d_i,RL_i);
            if (RL_LG_curr > ERL_LG_i) then
               ERL_LG_i ← RL_LG_curr;
               d_candidate ← d_i;
            end
         end
      end
   end
   return < d_candidate, ERL_LG_i >;
end
```

Fig. 3.6. ALGORITHM 3: Local assignment of task t within local group LG_i.

(set to 30%), so as to avoid overloading of the local groups. After that, in each of such local groups, including the one that originated the request, a candidate node that best performs the task is determined. Finally, among all the candidate nodes, the task is assigned to the one that ensures the highest network lifetime.

3.4 Energy-Aware Prototype

In an early version of this work [53] we implemented a prototype of the system. The prototype consists in a network of mobile devices composed of a set of HTC Hero smart phones and a set of Android emulators.
The prototype is general and the considered computational tasks may concern different application scenarios. Anyway, to support our experimental evaluation, we focused on data mining algorithms. In particular, we refer to the computational load as the energy consumption, a priori estimated, to execute a set of data mining tasks. These costs have been estimated through experimental evaluation.

As previously mentioned, the mobile nodes include a set of software components. These are designed according to the open/closed principle, so as to integrate new features without making any changes to existing ones. This section presents the design aspects and implementation details of the prototype software, describing its software architecture and implementation technologies. Moreover, the experimental evaluation of the system is shown.

```
Method globalAssignment
Input: task t, data set size s, set of neighboring clusters C

begin
    D ← ∅;
    foreach local group LG_i ∈ C do
        < d_candidate, RL_LG_i > ← CH_i.localAssignment(t, s);
        D.add(< d_candidate, RL_LG_i >);
    end
    ERL_net ← 0;
    d_best ← ∅;
    foreach node d_i ∈ D do
        RL_net ← estimateNetworkResidualLife(d_i, RL_LG_i);
        if (RL_net > ERL_net) then
            ERL_net ← RL_net;
            d_best ← d_i;
        end
    end
    CH_best ← getClusterHead(d_best);
    if (d_best = this) then
        startTask(t, s);
    else
        if (CH_best = this) then
            d_best.executeTask(t, s);
        else
            CH_best.requireTaskExecution(d_best, t, s);
        end
    end
end
```

Fig. 3.7. ALGORITHM 4: Global assignment of task t.

3.4.1 Implementation

The static model of the system has been represented by the class diagram of the modeling language UML. Through this diagram, shown in Figure 3.8, it was possible to specify the structural elements of the system and their static relationships, that is invariant over time. The most important classes are:

- *Node*. Represents a mobile device. It contains all the energy characteristics of the node and the features that allow to manage the life of the device in the system.
- *Task*. Represents a task. To support our experimental evaluation, we consider data mining tasks.
- *Cost Estimator*. Represents the EAS module of the scheduler that is responsible for estimating the energy characteristics of the device. The main functions are to verify if a device is eligible to perform a certain task and estimate the remaining life of the node, of the local group and of the network after the execution of a task.
- *Mapper*. This class is the heart of the EAS component. This includes the scheduling algorithm and a matchmaker. Allocate tasks, choosing the best node for tasks execution.
- *Scheduler*. The class guides scheduling activities. It receives requests to perform tasks and requires the allocation to the mapper.

- *ResourceMonitoring.* This class maintains information about the mobile node's resources such as available memory, CPU usage, and mainly, it monitors the battery level. It also estimates the computation and communication energy required to execute a specific tasks.
- *WEKA Classes.* This set of classes is the set of tasks that are submitted in the system, ie the data mining algorithms.

The operating system chosen for mobile devices of the prototype is *Android.* One of the key aspects in the implementation of the proposed architecture is the communication between the nodes. Android provides several ways to implement client-server applications, including *sockets.* The latter are used as the endpoint of inter-node communication flow across the MANET. Another important aspect concerns the execution and the type of tasks. Focusing on data mining tasks, it is investigated on which data mining algorithms a mobile node can invoke, which must be the size of the dataset, what type of result these algorithms return and in what format. To solve all these problems, we use the data mining algorithms provided by the popular software environment WEKA [58], exploiting the format that it uses for reading the data sets (i.e., ARFF files), and for the evaluation of results (e.g., the confusion matrix).

Android

Android OS is a Linux based operating system that is targeted for mobile, smart phone and tablet devices. It was developed by Google in cooperation with the Open Handset Alliance in November 2007. It is characterized by its openness and explicit features that enable developers to take full advantage of features of the hardware devices.
Android offers simple and powerful SDK that provide tools and different APIs for developing mobile apps in multi-platform environments. Android, in contrast to its rival Apples IOS (iPhone OS), can be deployed to a variety of hardware devices. As a result, a variety of handsets and hand held tablets running android OS , are distributed by different mobile carriers and manufacturers that share the majority of the global market. Google's new name for what we used to know as the Android Market, where Android users went to buy and download their mobile apps, is Google Play. *Google Play* was introduced in March 2012, by merging the Android market and Google Music services together. As a result, all android apps are now accessed from Googles multimedia content service together with music, books and etc.

The Android OS follows a fault-tolerant, secured and efficient design. It consists of Core applications like browser, contact and phone, an Application Framework that enables reuse and replacement of components; different libraries for multimedia, database, security and more; and a Linux kernel that is responsible for memory management and security issues. The main components of the android OS can be showed in different layers as shown in Figure 3.9.

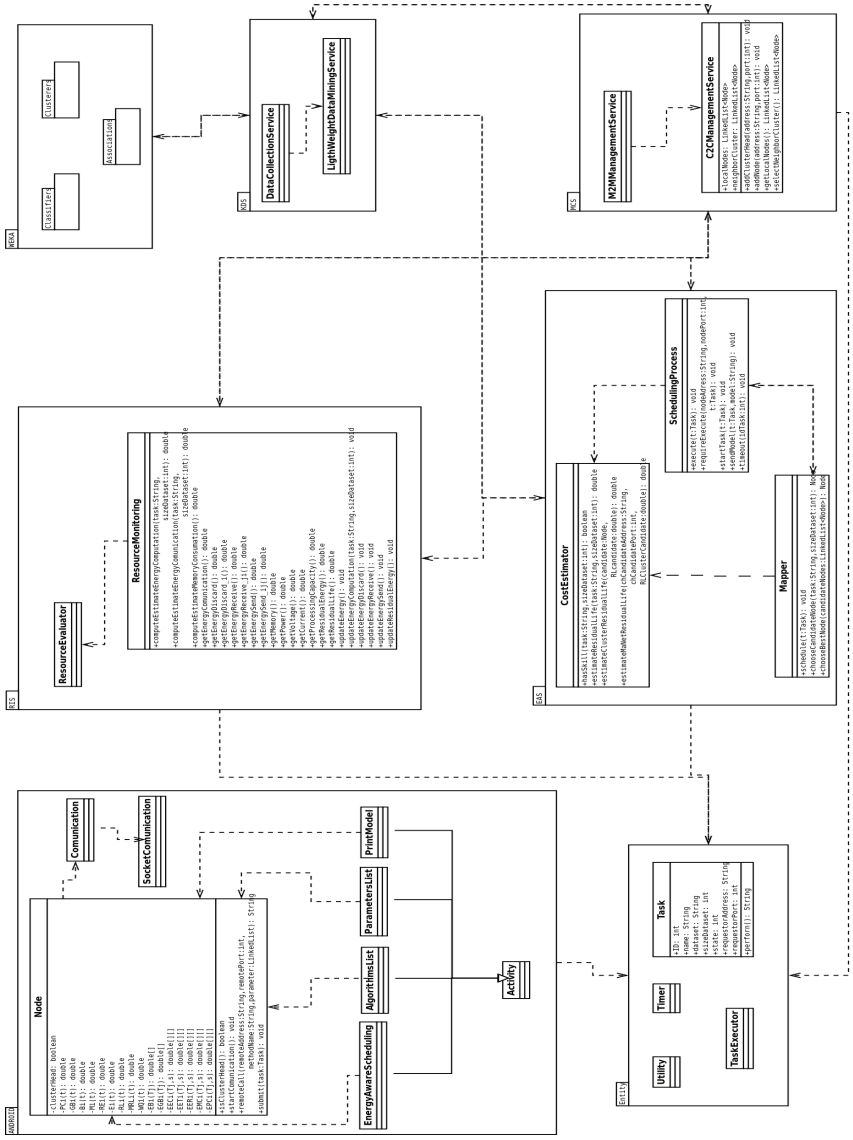


Fig. 3.8. Energy-aware prototype UML class diagram.

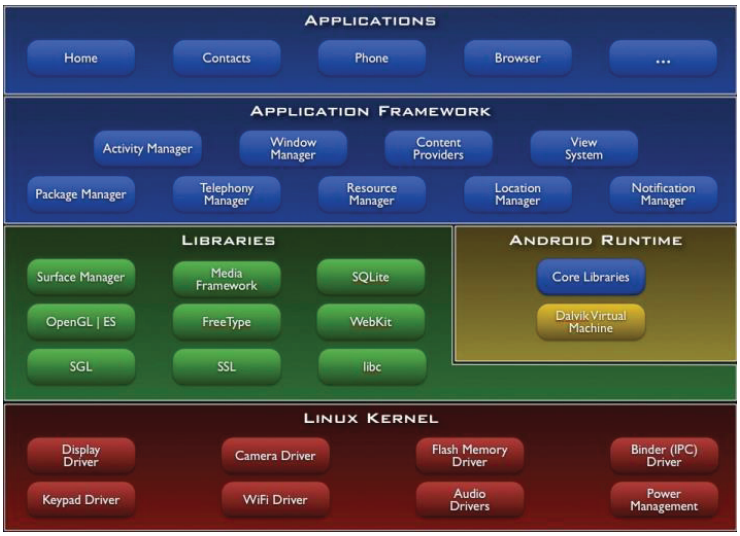


Fig. 3.9. Android Architecture.

Linux Kernel

This is part of the layer that is responsible for services like memory and process management, security, network stack, threading, camera, and audio drivers and more. It is also referred as an abstraction layer between the hardware and rest of the software stack. Android uses Linux 2.6 version for handling the above mentioned processes [59].

Android Runtime

The Dalvik Virtual machine and a set of core libraries that handle different functionalities found in core java libraries is included in this layer. Each android application runs separately on its own Dalvik virtual machine instance. The Dalvik Virtual Machine is a register based architecture that is designed for systems that are constrained in terms of memory and processor speed.

Libraries

Android contains a set of C and C++ libraries that are responsible for performance optimization and efficiency. This includes:

- *System C library* that is an implementation of the standard C system library (libc), tuned for embedded Linux-based devices;
- *Media Libraries* that support playback and recording of popular video and audio formats;
- *SQLite* that is a lightweight relational database engine;
- *LibWebCore*, a modern web browser engine;

- *SGL* a 2D graphics engine, 3D libraries based on OpenGL ES 1.0 APIs, Surface manager and more [59].

Application Framework

These are a set of managers that provide services for views, activities, content providers notifica-tions and more. Native libraries that are responsible for performance optimization are accessed through the application framework. Some of the components of the application framework layer are listed as follows:

- *Content Providers*, which manages access to a central repository data;
- *Resource manager*, that provides access to resources like graphics, layouts and more;
- *Notification Manager*, which is used for displaying custom alerts;
- *Activity manager*, that handles the lifecycle of applications and more.

Applications

These are a set of applications that are written in the Java programming language. Some of the examples include an email client app, SMS, browser, Contacts and more.
Android applications are written in the Java language, compiled into byte codes which will be converted to a .dex file (*Dalvik executable file*) using the *dx* converter. This will further be compiled in to android package file (*apk file*), that can be installed on the android devices. Taking in to account the multi user Linux system aspect of android OS, every Android application is treated as single Linux user. Each of the applications on the system is identified by a unique Linux user Id. Each process has also its own virtual machine.
Android applications are composed of one or more of the four basic application components. The main building blocks of an android application are listed and discussed as follows.

Activity. It is a core component of the android applications. An Activity represents a given task that can be done by the application. It can also be represented as a single screen User interface that interacts to do functions like dialing, taking photo and so on. A single android application is composed of multiple activities.

Services. Designed to keep running in the background, services are responsible for tasks like updates and remote processes. Services can be implemented for time taking operations and processes that need to be scheduled and carried out regularly. Services do not provide UIs.

Content Providers. These are used as interface to data. A content provider helps to maintain shared data between applications. Data can be shared and modified according to permissions using the content providers. Data encapsulation and data security is provided using content providers. Consistent and secured data access is managed and maintained using content providers and provider clients.

Broadcast receivers. These are broadcast announcements made by the system. Examples include a broadcast for announcing that battery is low or to inform an update is ready to be downloaded. Broadcast Receivers can be used to initiate services to do something.

One of the merits of developing for android is the cheap and easiness of the development environment. In addition to this, well organized documentations make it easy for developers to startup and dive in to the platform quickly. While developing android applications, Tools from the SDK can be invoked through command Line or the ADT. In this project, the Eclipse IDE is used for the application development .The ADT (*Android Development Tools*) is Eclipse plugin that is recommended for developing android applications since it offers direct invoking of tools during application development [59].

3.4.2 A case of study: Data mining tasks

Further to the specialization of the prototype for the execution of data mining tasks, it is necessary to evaluate the energy that they consume on mobile devices. As discussed in the previous section, the EA scheduler needs to know the estimated memory (EMC) and energy (EEC) consumption of the device candidate to run a task. We characterize the performance of a data mining algorithm running over a specific device and with respect to a given data set, in terms of the following metrics: memory consumption, battery depletion and execution time assuming that the data mining tasks are CPU-bound.

Given a task t_a and size s of the dataset to be analyzed, the EMC and EEC values for a given device d_i, $EMCi(t_a, s)$ and $EECi(t_a, s)$, are obtained on the basis of memory and energy consumption measurements performed on that device. Therefore, for each device, it is necessary to measure in advance the amount of memory and energy consumed by all the data mining tasks that can be executed on that device, to analyze datasets of different sizes.

We selected three data mining algorithms from the Weka library: (1) J48 for data classification, (2) K-means for data clustering, and (3) Apriori for association rules discovery. Table 3.2 reports memory and energy consumption of the three data mining algorithms on datasets of increasing size, based on their execution on a set of HTC Hero smart phones with Android OS. Each algorithm has been executed 10 times; the EMC and EEC values reported in the table are the average of the memory and energy consumed at the different execution. Two sample datasets, from the UCI KDD Archive, have been used as data sources: Covertype1 with J48 and US Census 902 with K-means and Apriori. The first dataset contains information about forest cover type for a large number of sites in the United States; the second one contains data extracted from the US Census Bureau Web site as part of the 1990 US census. As expected, the Table 3.2 shows that the energy consumption grows with the dataset size for all the data mining algorithms. For example, for J48, the EEC value ranges from about 178 to 2593 J, by increasing the dataset size from 0.1 to 0.8 MB. Similar trends can be observed also for the memory

consumption. For instance, for J48, the EMC value passes from about 19 MB with the dataset of 0.1 MB to about 28 MB with the dataset of 0.8 MB.

Table 3.2. EMC and EEC of three data mining algorithms on datasets of increasing size, based on their execution on a set of HTC Hero devices.

Technique	Algorithm	Dataset	Dataset size (MB)	EMC (MB)	EEC (J)
Classification	J48	Covertype	0.1	19.47	178.49
			0.2	20.15	396.94
			0.4	23.87	791.21
			0.8	27.68	2592.60
Clustering	K-means	US Census 90	0.1	16.73	33.97
			0.2	17.95	107.89
			0.4	19.72	251.75
			0.8	23.08	305.69
			1.6	26.40	575.42
Association rules discovery	Apriori	US Census 90	0.1	15.86	2.82
			0.2	16.97	5.91
			0.4	18.06	12.85
			0.8	19.87	35.96
			1.6	23.32	179.82
			3.2	26.92	310.36

3.4.3 System Evaluation

The mobile devices that compose the prototype are distributed over an area of 250,000 m^2, and are grouped into clusters. Tasks are generated on each node. The task arrival event is a Poisson process with variable frequency. The initial value of the transmission range was set to 100 meters and the initial energy level on each device is 24300 J. Each device was equipped with a network interface 802.11 b/g, with a bandwidth of 11 Mbps. The transmission energy is based on the model presented in Section 3.2.2.

In all the experiments, we refer to the computational load as the energy consumption, a priori estimated, to execute a set of data mining tasks (Table 3.2).

Since the goal of our scheduling policy is to maximize the energy levels of the nodes and, thus, extend the network lifetime, the simulation aims to study the behavior of the scheduler with respect to the energy depletion and network lifetime. Accordingly, we use as performance metrics the *residual life of the network*, the *number of alive devices* and the *number of completed tasks*. To assess the effectiveness of the proposed scheduler energy-aware (EA) we compared its performance with that achieved by the well known *Round Robin* (RR) scheduling algorithm. This comparison aimed at evaluating the throughput, in terms of number of completed tasks, and the energy consumption of

the two algorithms. Note that the comparison through the residual life parameter is significant only if the two algorithms execute the same number of tasks. Otherwise, the comparison can be made with respect to the number of alive devices and the number of completed tasks.

In a first experiment, the initial energy level of the devices in the network is on average the 75% of the peak value. The computational task used is the J48 data mining algorithm running over a data set of 800 Kbytes that takes 2 hours and 16 minutes to be executed, with an average energy consumption of about 2592 J. The total number of submitted tasks is 70; these tasks arrive to the system following a Poisson distribution with a frequency λ equal to 20 tasks per hour. The simulation time is not fixed a priori, since the experiment ends when all the scheduled tasks have been executed. The aim of this first experiment is to show that the proposed EA scheduler is effective in prolonging network lifetime compared to the RR algorithm. In particular, by executing the same number of tasks, the EA scheduler saves about 3 hours compared to RR (see Figure 3.10(a)).
The EA scheduler allocates only 55 tasks of the 70 submitted because recognizes that it is the maximum number of tasks that the devices in the network can execute. The EA scheduler completes 54 of such tasks because for the last allocated task the scheduled node, at execution time, has no longer the energy required to run it. Such node asks for the re-allocation of the task but no other nodes in the network have the energy needed to execute it. Thus, the task execution fails. The RR algorithm executes also 54 tasks, however, it is not aware of the energy availability of the nodes and, thus, allocates a number of tasks greater than the actual availability of the network. Consequently, it causes the turning off of 5 devices (see Figure 3.10(b)). We can conclude that the proposed EA algorithm allocates and completes the execution of all the tasks that can be executed over the network, prolonging the network lifetime by balancing the energy load and avoiding in such way devices turning off.

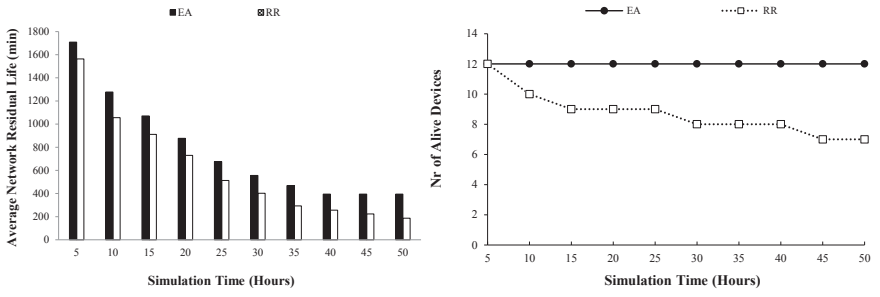


Fig. 3.10. (a)Average network residual life and (b) number of alive devices, after the execution of 54 tasks for both EA and RR algorithms w.r.t. simulation time.

In a second experiment the devices are less charged, with an average initial energy available of 30% of the peak value. The simulation time is not fixed, the experiment ends when all the scheduled tasks are executed; by fixing the number of submitted tasks to 15, this time is of 12 hours. Figure 3.11(a) and Figure 3.11(b) show, respectively, the number of alive devices and the number of completed tasks for both the EA and RR algorithms with respect to the simulation time. From the Figure 3.11(a) one can see that EA balances the energy load and remains with all the devices alive whereas RR assigns tasks also to device that do not have the necessary energy and, thus, causes the switching off of 6 devices. Moreover, at the end of the simulation time the EA algorithm completes 15 tasks versus 9 of RR (see Figure 3.11(b)). This means that when the tasks are compute demanding, compared to the overall network energy, the EA algorithm makes the best from the available energy outperforming RR also in terms of number of completed tasks.

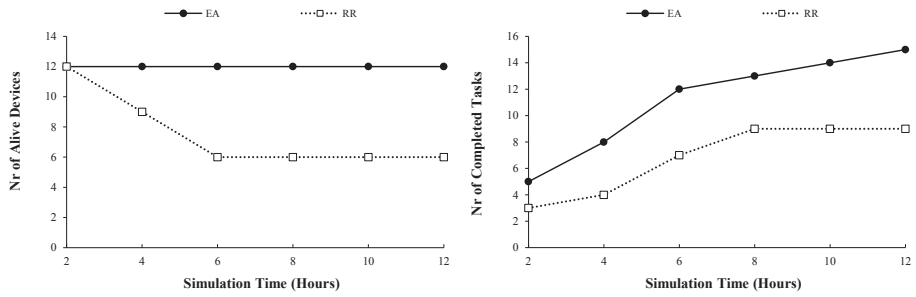


Fig. 3.11. (a)Number of alive devices and (b) number of completed tasks for both EA and RR algorithms w.r.t. simulation time.

We repeated the experiment by considering an increasing number of submitted tasks, ranging from 15 to 30. With the considered network energy configuration, the maximum number of J48 tasks, over a dataset of 800 KBytes, that can be executed are 17. Confirming the previous result, EA does not allocate a number of tasks greater than the maximum number supported by the network. It, thus, successfully executes all the allocated tasks (see Figure 3.12(b)). Conversely, RR allocates all the submitted tasks and, as shown in Figure 3.12(a), the number of devices turned on decreases by increasing the number of submitted tasks, reaching zero when 30 tasks are submitted.

In a scenario like the one considered, where the energy configuration of the different devices is rather heterogeneous, the energy load balancing effect of the EA is particularly significant. Figure 3.13 shows how the remaining energy is distributed among all the devices in the network. In particular, it is shown how this distribution changes by increasing the number of submitted tasks, for both the EA and RR algorithms. More precisely, it is specified the

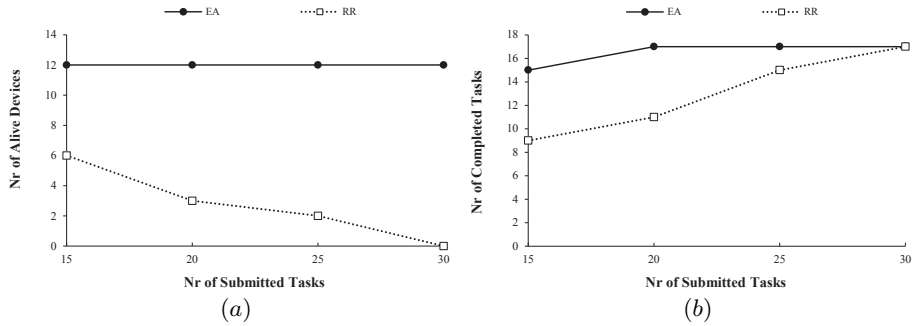


Fig. 3.12. (a)Number of alive devices and (b) number of completed tasks for both EA and RR algorithms w.r.t. the number of submitted tasks.

percentage of devices with (i) no energy ($RE = 0$), (ii) low remaining energy ($0 < RE \leq 1000$), (iii) medium remaining energy ($1000 < RE < 5000$) and (iv) high remaining energy ($RE \geq 5000$). One can note the effectiveness of the EA algorithm in balancing the energy level among the devices. In the case of the EA algorithm the percentage of devices with medium remaining energy is always rather high, regardless of the number of submitted tasks. Conversely, in the case of the RR algorithm, by increasing the number of submitted tasks the percentage of devices without energy grows always more, reaching the 100% when 30 tasks have been submitted and all the devices are turned off.

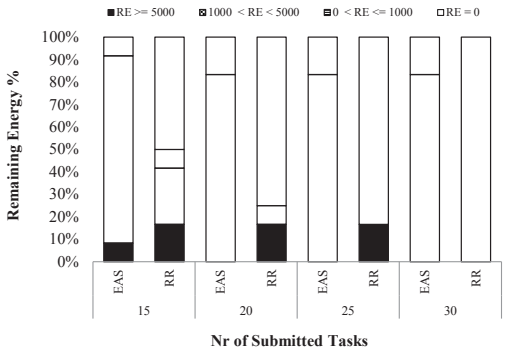


Fig. 3.13. Distribution of devices remaining energy for both EA and RR algorithms w.r.t the number of submitted tasks.

The efficiency of the EA scheduler is further confirmed when dealing with tasks of increasing computational load. Figure 3.14(a) and Figure 3.14(b) show that EA outperforms RR in terms of throughput and device lifetime. By increasing the computational load the EA scheduler completes a lower number of tasks because it allocates only the tasks that can be executed avoiding this way the switching off of the devices. Conversely, RR turns off a

number of devices that increases with the growing of the computational load. With a computational load of 2500 J, the two algorithms execute the same number of tasks but EA remains with all the devices switched on while RR turns off more than the 90% of the devices.

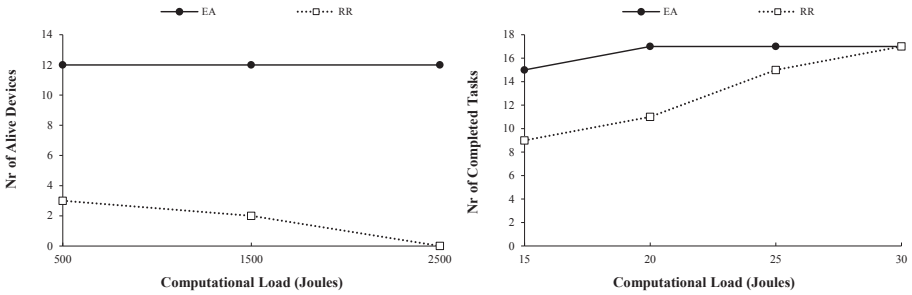


Fig. 3.14. (a)Number of alive devices and (b) number of completed tasks for both EA and RR algorithms w.r.t. the computational load.

In the last experiment four different network configurations are considered. For all the configurations the average initial energy level of the devices is the 30% of the peak value. The configurations differentiate only for the distribution of the energy among the devices. The EA scheduler with its dynamic strategy adapts its behavior to the different configurations always assigning the tasks to the most energy powerful devices. Differently, RR allocates the tasks without taking into account the energy availability of the devices. Thus, the performance of RR depends on the specific combination of devices and tasks: each time a task has to be allocated, RR efficacy depends on the charge of the device at the beginning of the scheduling queue and on the load of the task to be allocated. For example, if the most charged device is at the beginning of the scheduling queue when a high computational task has to be allocated, there is a chance of executing the task without turning off the device. Figure 3.15 shows that EA successfully executes all the submitted tasks in all the considered configurations (as EA performs in the same way in all the configurations, in Figure 3.15 we use the notation $EA_{[1,2,3,4]}$ to refer to EA in the 4 different configurations). The graph also shows the bad behavior of the RR: in all the considered configurations the throughput of EA is higher than the RR one. This is particularly evident in configuration RR_4 where the less charged devices are at the beginning of the scheduling queue. Consequently, a greater number of tasks have been allocated to such less energy powerful devices that are not able to execute them; for example when 5 tasks have been submitted, RR is unable to complete even a single task in correspondence of configuration RR_4.

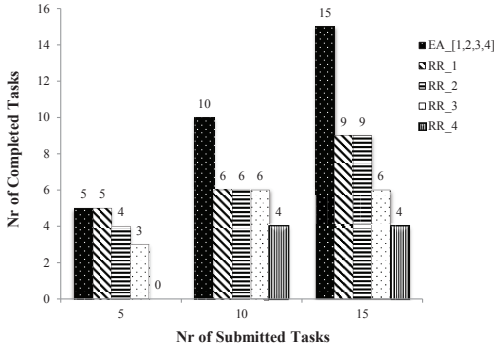


Fig. 3.15. Number of completed tasks in different energy configurations for both EA and RR algorithms w.r.t. the number of submitted tasks.

In general, the behavior of the RR scheduler which assigns tasks to each processors in equal portions and in circular order, lets its effectiveness depending on the combination task to be allocated and device at the beginning of the circular queue. For example, in the case of Figures 10 and 12 the curves representing the RR performance, are relative to the most convenient configuration for RR, the one where the most energy powerful devices are at the beginning of the circular queue. It is, thus, the configuration which allows to achieve the best performance with RR. Despite this, EA is much more efficient. If in those experiments we had considered one of the other possible configurations for RR, the improvement of our algorithm would have been even higher.

3.5 An Energy-Aware Simulator

The use of the energy-aware prototype does not allow to perform a meaningful experimental evaluation of the system due to the reduced number of nodes composing the MANET and the consequent low number of submitted tasks. Due to this, in recent work [55, 54, 56] we implemented a custom discrete-event simulator to evaluate the EA strategy considering more complex scenarios.

The goal of the evaluation is to assess whether the EA strategy is able to extend the network lifetime and the number of alive devices compared to related scheduling strategies, while meeting application-level performance constraints.

This section describes the design aspects, the implementation details of the simulator and the related scheduling strategies. Moreover a rich set of experiments are presented, considering both data mining tasks and generic tasks.

3.5.1 Implementation

The custom discrete-event simulator, models the operation of a energy-aware system as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur; thus the simulation can directly jump in time from one event to the next. The simulator is parametrized.

As a first step, the simulator builds a network of n mobile devices and let them grouping into clusters based on the algorithm described in [57]. Then, an initial energy capacity is assigned to each device. After the initial setup, the simulator generates a set of events by which each cluster member node get information about the residual energy of its cluster head, while each cluster head collect information about the energy status of its cluster and about the other cluster heads.
At this point, mobile devices start generating a set of tasks to be executed. Each task submission is an event in the system.
The simulator maintains a list of simulation events containing pending events. The pending events are the result of previously simulated event that have yet to be simulated themselves. The simulator processes the events in the event queue and stops when there are no more events to be processed, i.e. the event queue is empty.
Some of the major events that can be generated in the system are:

- *UpdateEnergyLevel* updates the energy information of each node.
- *DecreaseEnergyLevel* simulates battery consumption, taking into account the energy consumed for communication (ET) and energy consumed for the computation of a task (EC).
- *SubmitTask* is the arrival of a task in the system.
- *RequestTaskAllocation* is the request of task scheduling, done by a cluster member node to its cluster head.
- *AllocateTask* starts the scheduling strategy.
- *RequestLocalAssignment*, *LocalAssignment* and *ResultLocalAssignment* are used during the EA local assignment phase.
- *GlobalAssignment* and *RequestTaskAssignment* are used during the EA global assignment phase.
- *TaskAssigned* inserts a task in the task ready queue of the executor node.
- *TaskStarted* starts the execution of the task on the executor node.
- *TaskCompleted* ends the execution of the task.
- *TaskFailed* represents the failure of a task.
- *TaskResultDelivery* is the delivery of the output of a task to the requesting node.
- *TimerTask* occurs after a timeout within which a node wants to receive the result of a task.

An event is described by the time at which it occurs, the node that generates it and a type, indicating the code that will be used to simulate that event. When the simulator's event queue is empty, then all tasks are completed successfully, a large set of statistics is calculated (e.g., network residual life, network remaining energy, number of completed tasks, number of alive devices and so on) . These statistics are used for experimental evaluation of the system.

In Figure 3.16 is shown the flowchart that describe the simulation process, while in Figure 3.17 the UML class diagram of the simulator is presented.

3.5.2 Comparison Scheduling strategies

We compared the performance of the EA scheduler with that achieved by three related scheduling strategies:

- *Round Robin* (RR). RR assigns tasks to each device in equal portions and in circular order. Each cluster-head maintains a circular queue of local nodes and neighboring cluster-heads. When a task has to be allocated, the cluster-head assigns it to the current pointed node in the queue. The queue is scanned circularly, until at least one of the device is alive. RR does not make any check about energy requirements and availability, thus, a task is assigned to a device even if this does not have the required energy to execute it.
- *Smart Round Robin* (SRR). SRR is a modified version of RR that still maintains a circular queue composed of local nodes and cluster-heads of neighboring clusters. Differently from RR, SRR does take into account the residual energy of nodes to make scheduling decisions. Thus, SRR assigns a task to a node only if this has enough energy to run it.
- *Energy Consolidation* (EC). EC follows a two-phase allocation approach similar to that of the EA scheduler. When an assignment decision has to be made for a task, first the local phase is activated by allocating the task to the most discharged node in the cluster, among the ones having enough energy to run it. If there are no local nodes that can run the task, the global phase is activated and the task is assigned to the most discharged node among the neighboring clusters, after having verified that the node has enough energy. The main difference between EA and EC is in the optimization function. EA maximizes the number of alive devices prolonging the network lifetime by balancing the energy load among the devices. Conversely, EC consolidates the devices by filling first the least loaded ones, thus avoiding energy fragmentation in the network.

For each of the scheduling strategies considered (EA, RR, SRR and EC) we implemented three different ready queue management policies: (i) *First In First Out* (FIFO), where processes are assigned the CPU in the order they request it; (ii) *Shortest Job First* (SJF), that assigns the CPU to the process

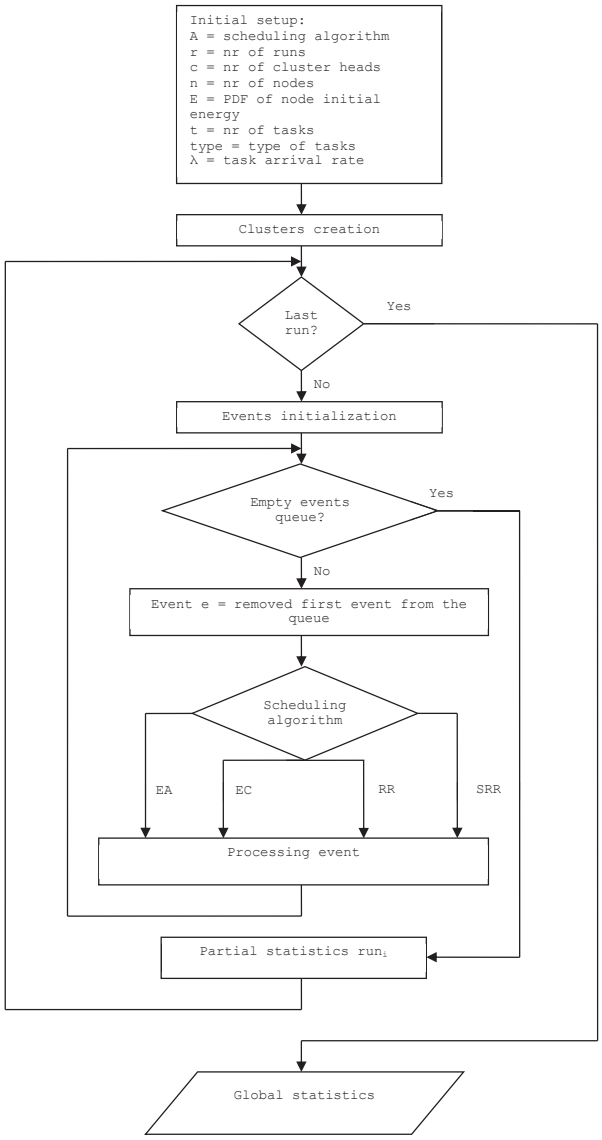


Fig. 3.16. Discrete-event simulator flowchart.

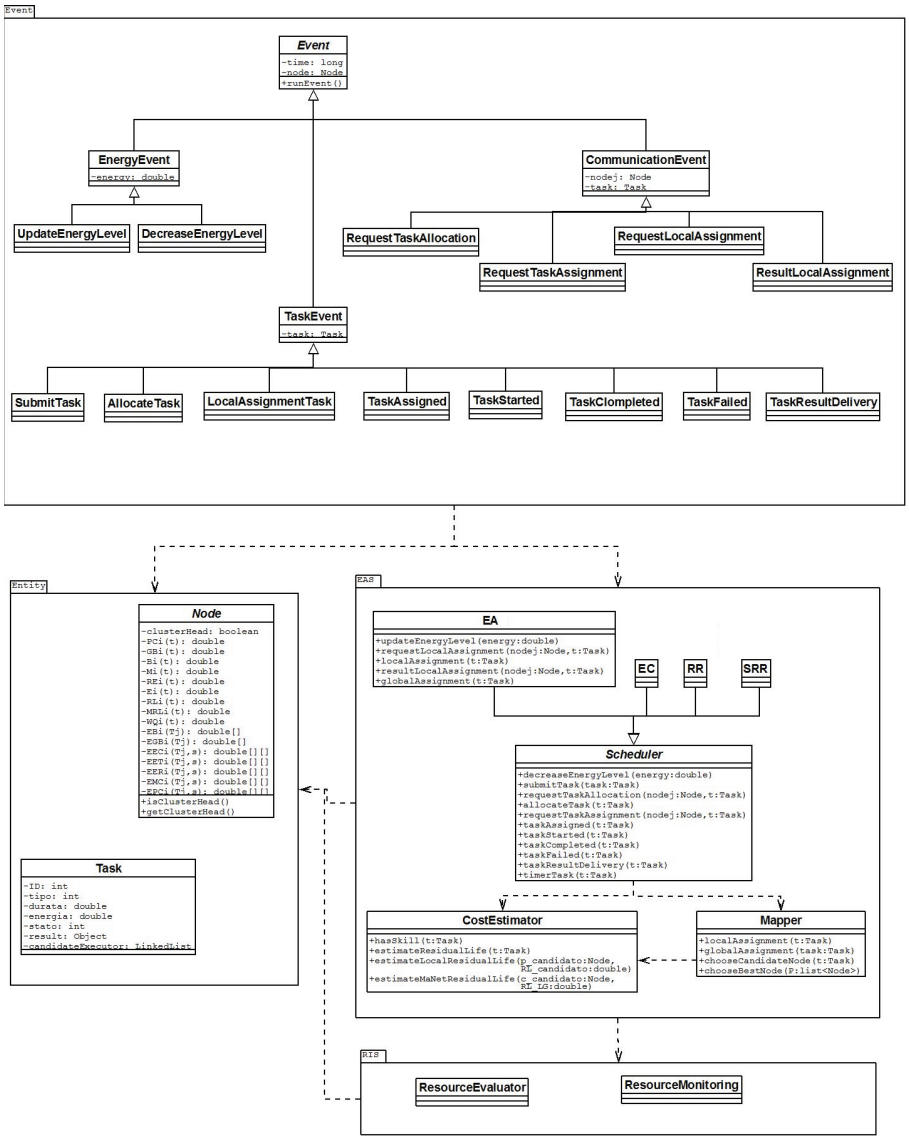


Fig. 3.17. Discrete-event simulator UML class diagram.

that has the smallest next CPU burst; and (iii) *Completely Fair Scheduler* (CFS) that assigns the CPU to the process that has been waiting for the CPU for the largest amount of time. The default ready queue management policies is the FIFO policy.

CFS is the preemptive scheduling algorithms adopted by Linux 2.6.23 and used in Android mobile devices. CFS tries to give tasks a fair amount of the processor time. When one or more tasks are not given a fair amount of time, then time to execute is assigned to out-of-balance tasks. To determine the balance, CFS maintains the amount of time assigned to each task in the so called *virtual runtime*. The smaller is the virtual runtime of a task, the smaller is the amount of time a task has been permitted access to the processor, while the higher its need for the processor.

3.5.3 System Evaluation

We evaluate the EA scheduling strategy considering both data mining tasks and generic tasks. To the scope of our experimental evaluation the simulator instantiates 100 mobile devices distributed over an area of 250,000 m^2. As a result of the clustering procedure, 20 devices act as cluster-heads, with an average cluster size of 5. An initial energy capacity ranging from 3,000 J to 11,000 J is assigned to each device, following a normal distribution.

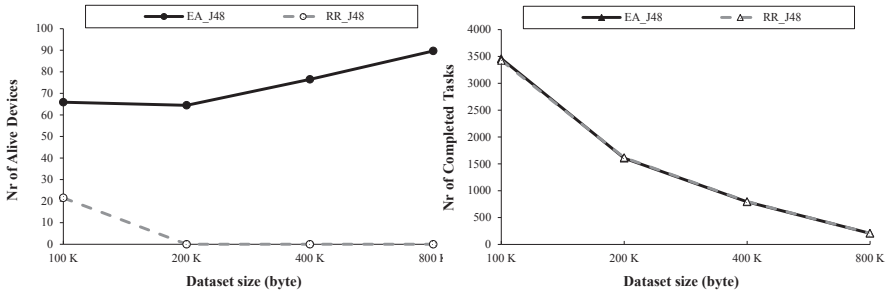
Data mining tasks

We focus on data mining tasks as collaborative data analysis represents a quite general example of application scenario. The EMC and EEC values reported in Table 3.2 have been used as input for the simulations.

The simulation aims at studying the behavior of the scheduler with respect to the energy depletion and network lifetime. Accordingly, as performance metrics, we use the number of alive devices, the number of completed tasks, and the network residual life at the end of the simulation. To assess the effectiveness of the EA strategy, we compared its performance with the one achieved by round-robin (RR) scheduling algorithm.

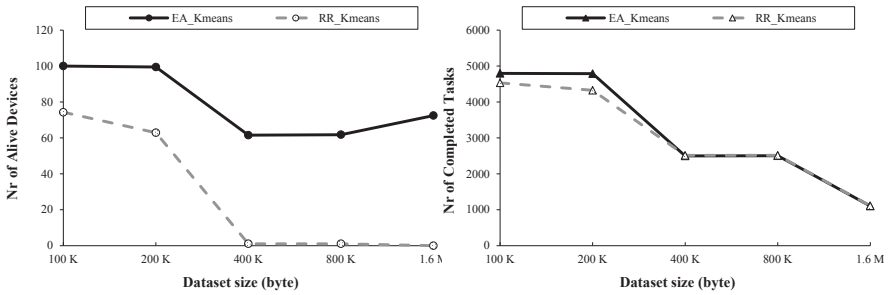
In a first set of experiments, for each reference algorithm (J48, K-means, and Apriori), we ran a set of tasks by varying the size of the dataset to be mined from 100 kB to 3.2 MB. Each simulation lasts 30 hours, with tasks that arrive following a Poisson distribution with a frequency $\lambda = 160$ tasks/hour. Figures 3.18(a), 3.19(a) and 3.20(a) show the number of alive devices at the end of the simulation for J48, K-means, and Apriori, respectively, using the EA and RR strategies. With all data mining algorithms and dataset sizes, the number of alive devices with EA is greater than (and in a few cases equal to) that of RR. In particular, Figures 3.18(a) and 3.19(a) show that there are no alive devices with RR for datasets greater than 200-400 kB using J48 and K-means. In contrast, in the same configurations, EA keeps alive a high percentage of the devices. Additionally, we can note that, with EA, the number

of alive nodes increases with the dataset size. This is due to the fact that, the greater the dataset size, the greater the energy required to complete the task. Since the EA scheduler does not allocate tasks when the available devices do not have enough energy, this leads to a higher number of alive nodes. It is important to note that the higher number of alive devices ensured by EA compared to RR, is obtained without reducing the number of tasks completed, as shown in Figures 3.18(b), 3.19(b) and 3.20(b) . Figure 3.21(a) shows the network residual life measured at the end of the experiments. The figure confirms that the EA scheduler is effective in prolonging network lifetime compared to the RR algorithm.



(*a*) (*b*)

Fig. 3.18. (a) Number of alive devices and (b) Number of completed tasks w.r.t. dataset size, using EA and RR with J48.



(*a*) (*b*)

Fig. 3.19. (a) Number of alive devices and (b) Number of completed tasks w.r.t. dataset size, using EA and RR with Kmeans.

In a second set of experiments, for each reference algorithm (J48, K-means, and Apriori), we ran a set of tasks with a fixed dataset size (200 kB) but with task arrival rate λ varying from 80 to 1280 tasks per hour. Figures 3.21(b)

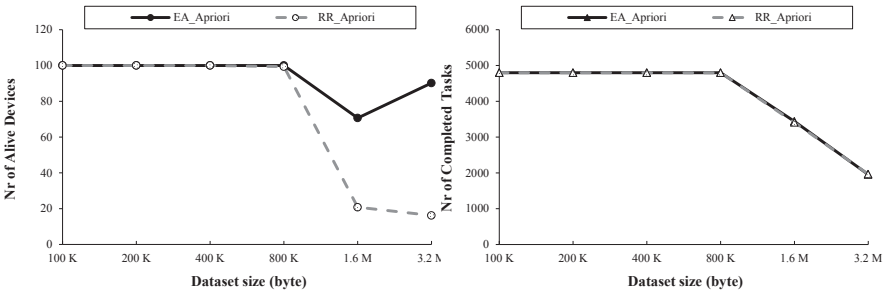


Fig. 3.20. (a) Number of alive devices and (b) Number of completed tasks w.r.t. dataset size, using EA and RR with Apriori.

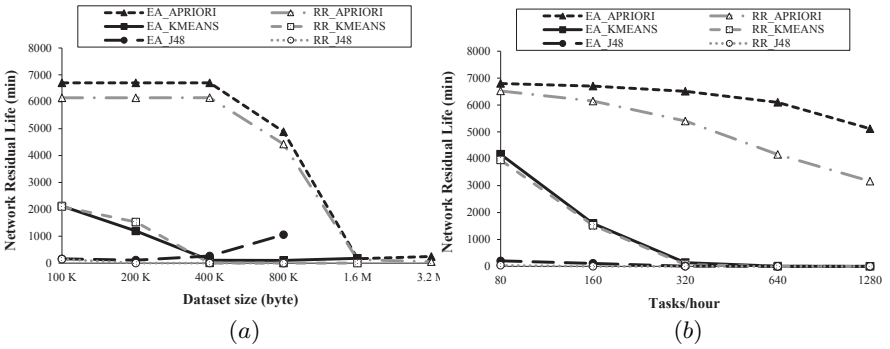


Fig. 3.21. Network residual life using EA and RR with Apriori, Kmeans and J48, w.r.t. (a) dataset size; (b) task arrival frequency.

shows the network residual life measured at the end of the experiments for the three algorithms, using EA and RR. As expected, increasing the task arrival rate, the network residual live tends to zero both for EA and RR. However, for the lightest of the three data mining algorithms (Apriori), the residual life does not reach zero and the difference between EA and RR increases with λ in favor of EA. Figure 3.22(a) shows the number of alive devices for the three algorithms, using EA and RR. The results demonstrate, also in this case, that the number of alive devices with EA is greater than that achieved by RR. Figure 3.22(b) compares the performance of EA and RR in terms of completed tasks for the three algorithms. With Apriori, both EA and RR are able to complete more tasks as λ increases, but EA ensures better performance. With J48 and K-means, with a given task arrival rate, the number of completed tasks cannot increase because the network residual life is zero, as shown in Figure 3.21(b). However, even in these cases, there is a slight advantage for EA compared to RR with $\lambda < 320$ tasks/hour.

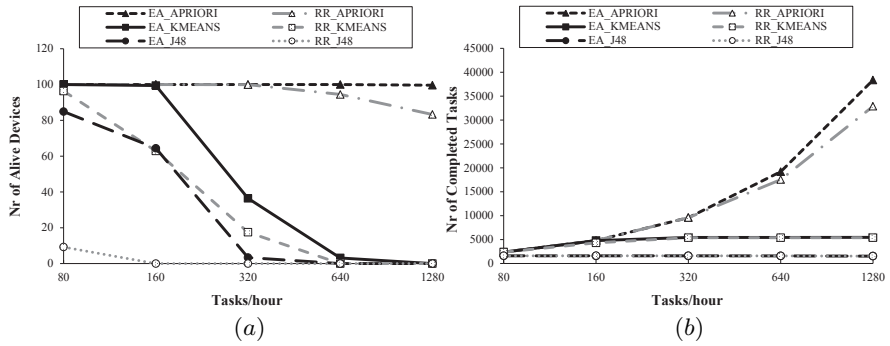


Fig. 3.22. (a) Number of alive devices and (b) Number of completed tasks w.r.t. task arrival frequency using EA and RR with Apriori, Kmeans and J48.

We also evaluated the average waiting and completion times of the three algorithms using EA and RR. Figures 3.23 shows that waiting and completion times increase with the task arrival rate for all the algorithms, with both EA and RR. With Apriori and K-means the difference between EA and RR is marginal for all the task arrival rates, while with J48 the difference increases significantly only for $\lambda > 320$ tasks/hour. These results show that EA ensures energy efficiency without limiting the performance execution.

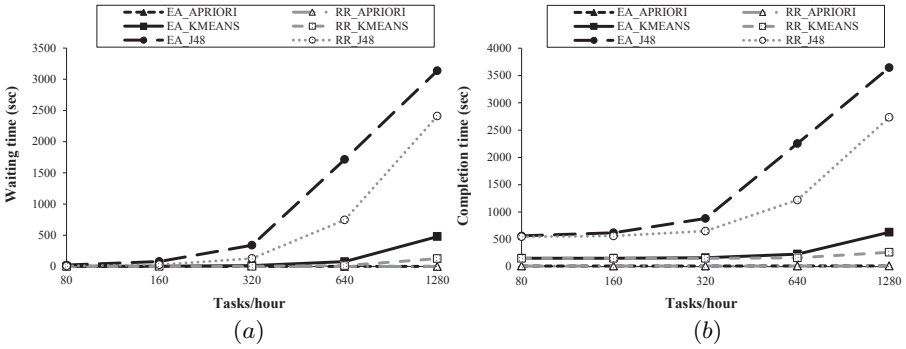


Fig. 3.23. (a) Waiting time and (b) Completion time w.r.t. task arrival frequency using EA and RR with Apriori, Kmeans and J48.

Finally, we evaluated how the number of alive devices and the number of completed tasks is affected by the distribution of the initial energy level among nodes. In particular, we considered four energy configurations:

- $CONF_1$: the energy is assigned to the devices following a normal distribution with average at 40% of the peak value (this is the default configuration used also for the experiments presented above).

- $CONF_2$: all the devices start with the same energy level, equal to 40% of the peak value.
- $CONF_3$: the energy is uniformly distributed in the range 15%-65% of the peak value.
- $CONF_4$: the energy is assigned using a bimodal distribution, with local maxima at 30% and 75% of the peak value.

For all the configurations, the total energy level available is constant and equal to about 700 kJ. The algorithm considered for this evaluation is J48 on a data set of 200 kB. The results, obtained after the submission of 2500 tasks with $\lambda = 80$ tasks/hour, are shown in Figure 3.24. Figure 3.24(a) shows that EA keeps alive almost the same number of devices in all the configurations. On the contrary, since RR assigns tasks to each device in equal portions and in circular order, without taking into account their remaining energy, its effectiveness depends on the combination of task to be allocated and device at the beginning of the circular queue. In particular, we observe that the effectiveness of RR is comparable with that of EA only when all the devices have the same level of energy ($CONF_2$), which however is not a real scenario as devices are usually heterogeneous in terms of energy availability. Figure 3.24(b) shows that EA completes almost the same number of tasks in all the considered configurations, while in some cases the number of completed tasks with RR varies significantly (see for example $CONF_3$ versus $CONF_4$)

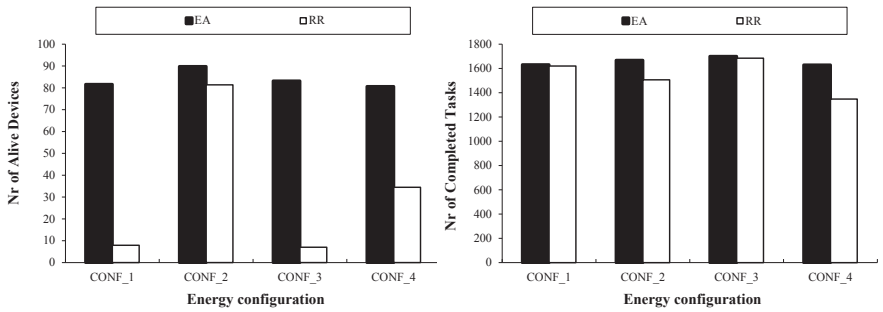


Fig. 3.24. (a) Number of alive devices and (b) Number of completed tasks w.r.t. energy configuration using EA and RR with J48.

As a concluding remark, the experimental results discussed above demonstrated that a significant improvement can be achieved using EA compared to the RR in a distributed mobile scenario. In particular, the EA strategy: i) resulted effective in prolonging network lifetime by reducing the energy consumption, without restraining the number of completed data mining tasks; ii) in all the experiments performed, it was able to keep alive most of the mobile devices thanks to its energy load balancing strategy.

Generic tasks

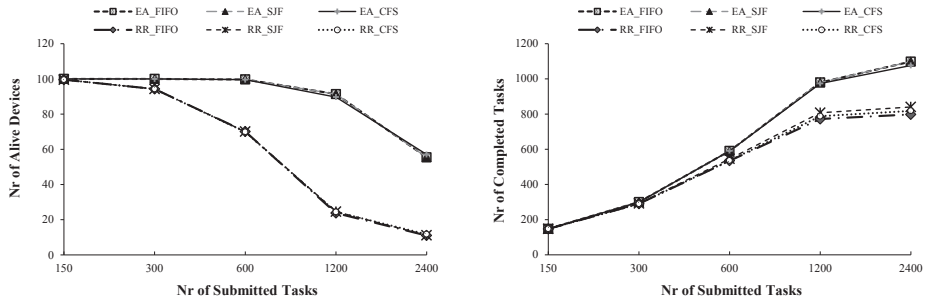
The following set of experiments considers generic tasks. Each task is characterized by the amount of energy required for its execution, ranging from 30 to 3000 J, and the execution time, ranging from 45 seconds to 160 minutes. Both the amount of energy and the execution time follow a Zipf distribution with skewing factor equal to 0.8.
Unless otherwise specified, the task arrival event is a Poisson process with a rate λ ranging from 80 to 2400 tasks/hour.

Since the goal of the EA scheduling policy is to maximize the number of alive devices and the network lifetime, we evaluated how effective is the EA strategy in achieving these goals compared to the related scheduling strategies introduced above. As the comparison in terms of network lifetime is significant only if the compared schedulers execute exactly the same tasks, we focus in particular on the following metrics, which allow to highlight the performance even if the compared schedulers do not execute exactly the same tasks: (i) the *number of alive devices*, (ii) the *number of completed tasks*, and (iii) the amount of useful work performed by the devices that we refer to as *workload*. The workload metric has been introduced to take into account the fact that tasks are heterogeneous. In fact, because of task heterogeneity, using only the number of alive devices and the number of completed tasks as performance metrics is not enough to evaluate the actual computation carried out by the devices.

The aim of this first set of experiments is to compare the performance of the EA task allocation scheme versus EC, RR and SRR with respect to the number of submitted tasks. We consider an increasing number of submitted tasks, from 150 to 2400, that arrive following a Poisson distribution with frequency $\lambda = 80$ tasks per hour. All the allocation strategies are evaluated considering the three ready queue scheduling policies introduced earlier: FIFO, SJF and CFS.

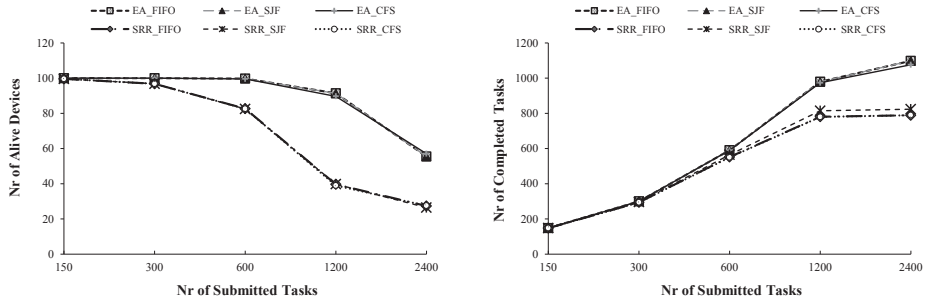
Figures 3.25(a) and 3.26(a) compare the number of alive devices of EA, respectively, with those of RR and SRR. As expected, the number of alive devices decreases with the number of submitted tasks, while the number of completed tasks increases. Nonetheless, with EA the number of alive devices decreases significantly only after the submission of 1200 tasks. Regardless of the ready queue scheduling policy, the number of alive devices with EA is greater than those of RR and SRR, and the advantage of EA increases with the number of submitted tasks. Similarly, EA completes a higher number of tasks than those of RR and SRR as can be noticed in Figures 3.25(b) and 3.26(b). This trend is even more evident for a number of submitted tasks greater than 600.

The same holds in the comparison with EC (Figures 3.27(a) and 3.27(b)). In contrast to the RR and SRR strategies, EC behaves slightly different on the basis of the ready queue scheduling policy used. More precisely, when EC uses the SJF policy, it improves the number of completed tasks that however is far



(a) (b)

Fig. 3.25. (a) Number of alive devices and (b) Number of completed tasks w.r.t. submitted tasks, using EA and RR with FIFO, SJF and CFS policies.



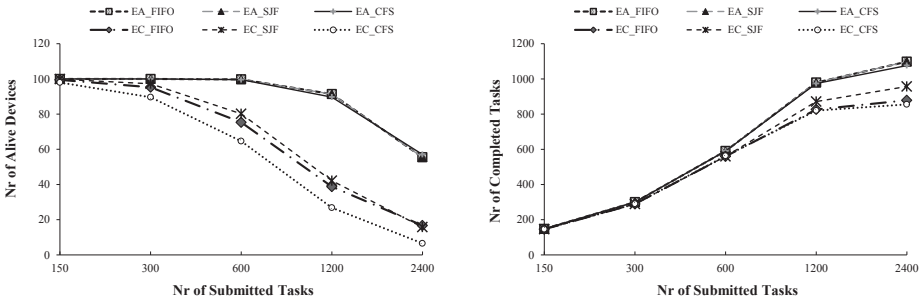
(a) (b)

Fig. 3.26. (a) Number of alive devices and (b) Number of completed tasks w.r.t. submitted tasks, using EA and SRR with FIFO, SJF and CFS policies.

from the number achieved by EA. On the contrary, when EC uses the CFS strategy it gets worse in terms of alive devices, reaching the same bad results as RR and SSR. The difference in the number of alive devices and executed tasks between EA and EC can be explained as follows. EC consolidates the devices, so the alive devices are more loaded compared to those of EA and are able to execute heavier tasks. As EA balances the energy load among the devices in the network, it remains with more devices alive that are less charged than those of EC. Accordingly, we expect that in terms of workload the two strategies present more or less the same performance.

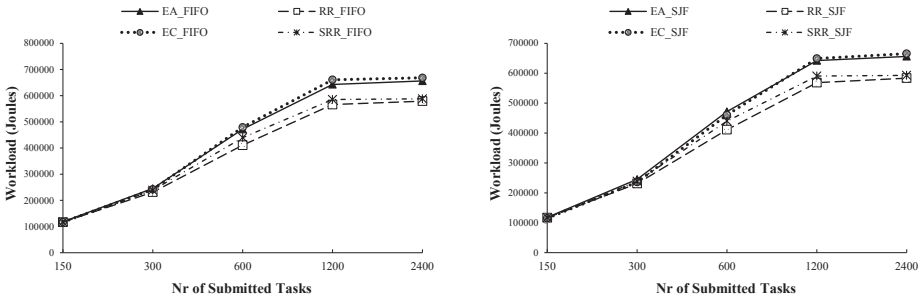
Figures 3.28 and 3.29 show that for very low number of submitted tasks (around 330) all the strategies with all the policies (SJF, FIFO and CFS) present almost the same behavior. By increasing the number of submitted tasks we can see that EA and EC outperform both RR and SRR carrying the same workload in case of the SJF and FIFO policies, whereas in case of the CFS policy EA performs significantly better than EC.

As shown in the previous figures, EA outperforms the other strategies both in the number of alive devices and in the number of completed tasks.



(a) (b)

Fig. 3.27. (a) Number of alive devices and (b) Number of completed tasks w.r.t. submitted tasks, using EA and EC with FIFO, SJF and CFS policies.



(a) (b)

Fig. 3.28. Workload with (a) FIFO and (b) SJF policies w.r.t. the number of submitted tasks, using EA, RR, SRR and EC.

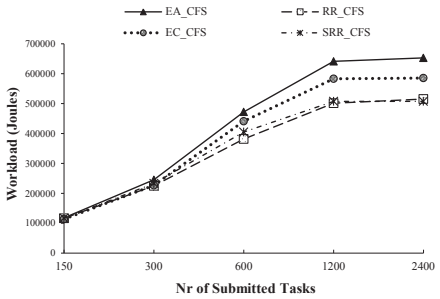


Fig. 3.29. Workload with CFS policy w.r.t. number of submitted tasks, using EA, RR, SRR and EC.

This is mainly due to the fact that EA does not allocate all the submitted tasks but only those that can be executed by the network. In fact, as we can notice in Figure 3.30, EA completes 100% of the allocated tasks, whereas RR the 75% (almost the same for all the scheduling policies) and, finally,

EC completes 80% of the allocated tasks that reaches the 95% using the CFS policy. The behavior of EC depends on the fact that this strategy consolidates the most powerful devices, discharging those with lower energy. Thus, during the assignment process it may happen that EC leaves the devices to which a task has been assigned with an energy load just enough to execute the just assigned task. For this reason, it might happen that some of these devices consume a small amount of energy for communication before the execution of the task, resulting in the inability of executing the assigned task.

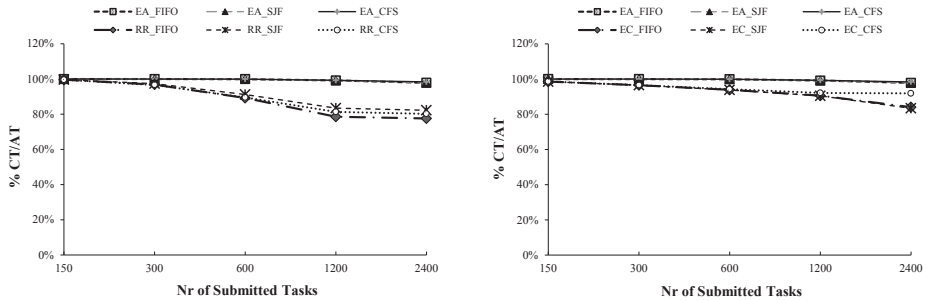


Fig. 3.30. Ratio between completed tasks (CT) and assigned tasks (AT) w.r.t. submitted tasks: (a) EA vs RR, and (b) EA vs EC, with FIFO, SJF and CFS policies.

As the devices are heterogeneous in terms of initial energy level, the energy load balancing effect of EA is particularly effective. Figure 3.31 shows how the remaining energy is distributed among all the devices in the network. In particular, it is shown how this distribution changes by increasing the number of submitted tasks, for EA, RR, SRR and EC. More precisely, it is specified the percentage of devices with (i) no remaining energy ($RE = 0$), (ii) low remaining energy ($0 < RE <= 1000$) (iii) medium remaining energy ($1000 < RE < 5000$) and (iv) high remaining energy ($RE >= 5000$). We report only on the results of the FIFO policy because they are similar to those obtained with SJF and CFS. One can note the effectiveness of the EA algorithm in balancing the energy level among the devices: the percentage of devices with medium remaining energy is always rather high, regardless of the number of submitted tasks. In particular, up to 600 tasks, 85% of devices remains with high energy values and up to 1200 tasks it keeps alive 95% of devices. Conversely, with RR, SRR and EC schedulers, when the number of submitted tasks increases the percentage of devices without energy grows always more, reaching 90% with RR, 75% with SRR, and 83% with EC after the submission of 2400 tasks.

In the remaining experiments we do not report on results of the CFS scheduling policy. This is because all the considered tasks have the same pri-

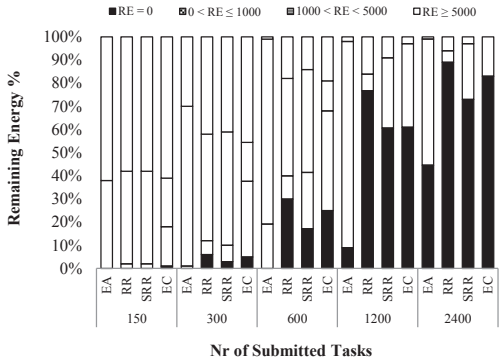


Fig. 3.31. Distribution of remaining energy using EA, RR, SRR and EC w.r.t. number of submitted tasks.

ority. Thus, we could not exploit the rationale of CFS that is essentially based on task priorities. Moreover we do not show other results concerning SRR, because we noted that its behaviour is almost the same as RR.

In the second set of experiments we compared the performance of the EA scheduler versus EC and RR considering tasks of increasing computational load, ranging from 250 to 4000 J. We set the number of submitted tasks to 2500, that arrive with a frequency $\lambda = 80$ tasks per hour. At each run, 2500 homogeneous tasks are submitted, all having the same computational load.

By increasing the computational load EA, EC and RR complete a decreasing number of tasks. Figure 3.32(b) shows that all the related schedulers complete the same number of tasks, but since EA and EC allocate only the tasks that can be executed, they avoid switching off the devices. Conversely, RR turns off a number of devices that increases always more till reaching 100% when the computational load is greater than 500 J, as shown in Figure 3.32(a). Differently, for EA and EC the number of alive devices increases with the computational load. This is because a lower number of tasks has been completed, and, thus, the alive devices remain more charged. For the same reason, network lifetime increases with the computational load for EA and the EC, whereas it decreases for RR until reaching zero, as shown in Figure 3.33.

The aim of this third set of experiments is to compare the performance of EA versus RR and EC considering task arrival rates ranging from 80 to 1280 tasks per hour.

Before presenting the experimental results, we briefly outline the expected behavior of the local scheduling policies when the submission frequency increases. By increasing the task arrival frequency, the devices consume more communication energy, so less energy remains for executing tasks for all the strategies. This is particularly evident using the FIFO policy, where the number of completed tasks decreases with the task arrival frequency, conversely to what happens with the SJF policy where the number of completed tasks

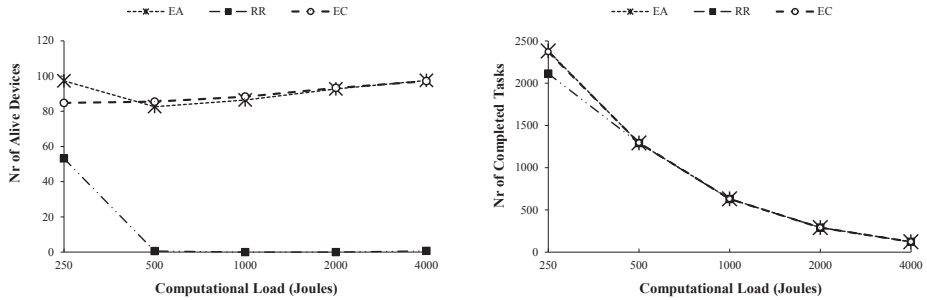


Fig. 3.32. (a) Number of alive devices and (b) Number of completed tasks w.r.t. computational load, using EA, RR and EC.

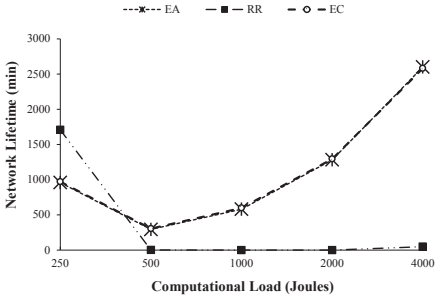


Fig. 3.33. Network Lifetime using EA, RR, SRR and EC w.r.t computational load.

increases. The reason of this can be explained as follows. By increasing the task arrival frequency, a larger number of lightweight tasks are submitted. This happens because we use a Zipf distribution of the energy required to execute the tasks, therefore the number of lightweight tasks is higher. In this condition, the SJF policy assigns the CPU to the most lightweight task moving it before all the preceding heavier tasks in the ready queue. This way, SJF increases the number of completed small tasks. RR is the scheduler that takes more advantage from the SJF policy because it allocates all the submitted tasks even if not all of them can be executed by the devices in the network. In contrast, EA and EC do not allocate a task if none of the available nodes satisfy the required energetic constraints, as will be shown in Figure 3.35(a).

Let's first discuss the case where all the strategies use the SJF policy. Figure 3.34(a) shows that the number of alive devices for EA decreases with the task arrival frequency up to 320 tasks per hour, while, for higher frequency values the number of alive devices decreases more slowly until remaining almost constant. This is because EA completes less tasks, as can be noted from the graph in Figure 3.34(b). For what concerns EC, for frequency values lower than 320 tasks per hour, it maintains alive a considerable lower number of devices compared to EA, as expected; for higher values the two strategies keep

the same number of alive devices. Despite this, we can see in Figure 3.34(b) that the number of completed tasks for EC is always lower than that of EA. RR remains with the lowest number of alive devices and, in particular for frequency values greater than 320 all the devices are turned off.

As said before, RR with SJF is the only strategy that takes advantage from the increasing frequency. As such, as can be noted from Figure 3.34(b), for frequency values higher than 320, the number of tasks completed by RR is considerably higher than those of both EA and EC. As stressed above, the advantage of RR in the number of completed tasks is due to the fact that RR allocates all the submitted tasks even if not all of them can be executed by the devices. Figure 3.35(a) shows that, in contrast, EA completes always almost all the allocated tasks down to 90% only for very high frequency values. EC completes a number of tasks ranging from 80 to 85%. In contrast, the number of tasks completed by RR decreases significantly with the frequency, reaching 40% with a frequency of 1280 tasks per hour. Thus, the advantage in terms of number of completed tasks of RR is due to the fact that it executes more lightweight tasks. This is confirmed by the amount of workload performed by the three reference strategies. In Figure 3.35(b) one can see that EA bears a higher workload. Only for very high frequency values, around 1000 tasks per hour, the workload of RR and EA strategies becomes the same, whereas, for frequency values greater than 320, EC behaves like EA. Furthermore, we highlight here that RR obtains this high number of completed tasks sacrificing the devices that at the end are all turned off.

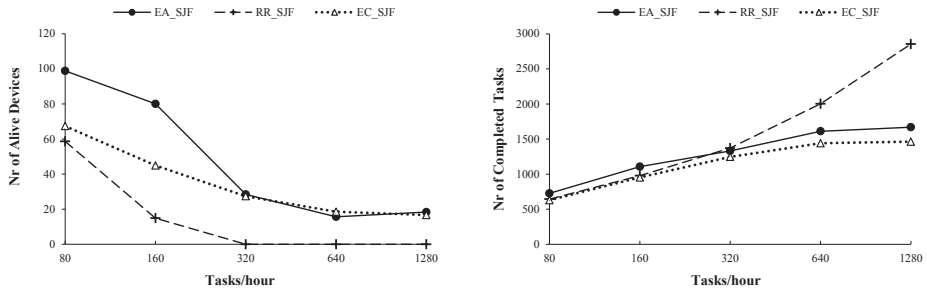


Fig. 3.34. (a) Number of alive devices and (b) Number of completed tasks w.r.t. task arrival frequency, using EA, RR and EC with SJF policy.

Figure 3.36(a) shows the number of alive devices of EA, RR and EC when the FIFO scheduling policy is used. As expected, the number of alive devices for all the strategies decreases with the tasks arrival frequency. Once again EA is able to keep the higher number of alive devices for frequency values lower than 320. For higher values, EC maintains a slightly larger number of alive devices. This is mainly due to the fact that, as one can see from Figure 3.36(b),

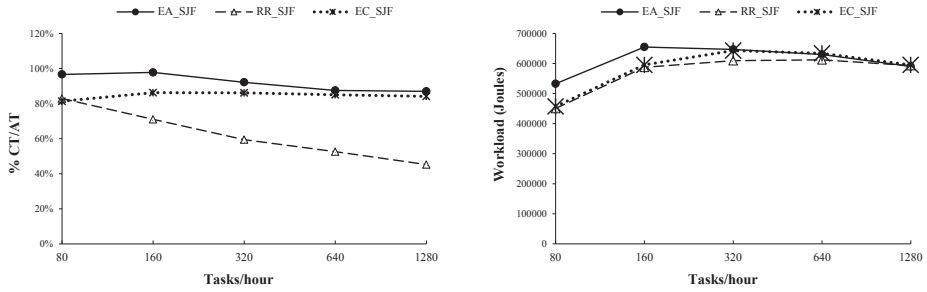


Fig. 3.35. (a) Ration between completed tasks (CT) and assigned tasks (AT) and (b) Workload w.r.t. task arrival rate using EA, RR and EC with SJF policy.

EC, mostly for frequency values greater than 320, executes a number of tasks much lower than EA. RR once again for frequency values greater then 320 switches off all the devices. However, differently from what happens with the SJF policy, here RR completes a significantly lower number of tasks than EA as reported in Figure 3.37(a). This figure shows that: (i) EA completes always almost all the allocated tasks dropping to 90% only for very high frequency values; (ii) EC completes a number of tasks ranging from 80 to 85%; (iii) the number of tasks completed by RR is always quite low until falling to 5% with a frequency of 1280 tasks per hour. Even more, the workload of RR is always substantially lower than EA, while the workload sustained by EC equals the one of EA for frequency values greater than 320 (see Figure 3.37(b)).

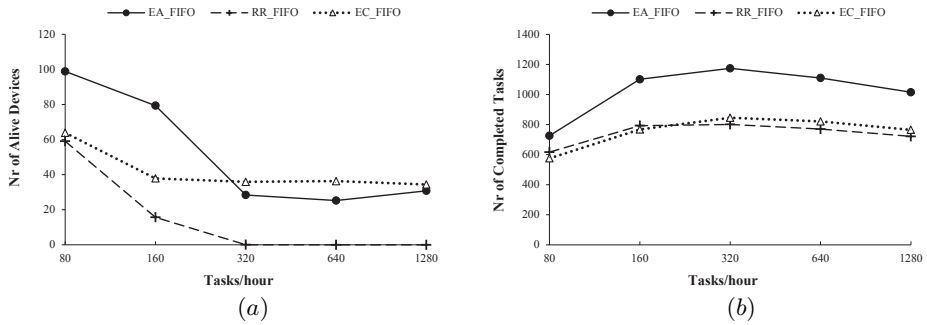


Fig. 3.36. (a) Number of alive devices and (b) Number of completed tasks w.r.t. task arrival rate, using EA, RR and EC with FIFO policy.

Figures 3.38(a) and 3.38(b) show the waiting time and the completion time of the tasks for all the three allocation schemes. For the SJF case, both the waiting time and the completion time of EA and RR are very similar: RR is slightly better only for frequency values greater than 640. The EC times

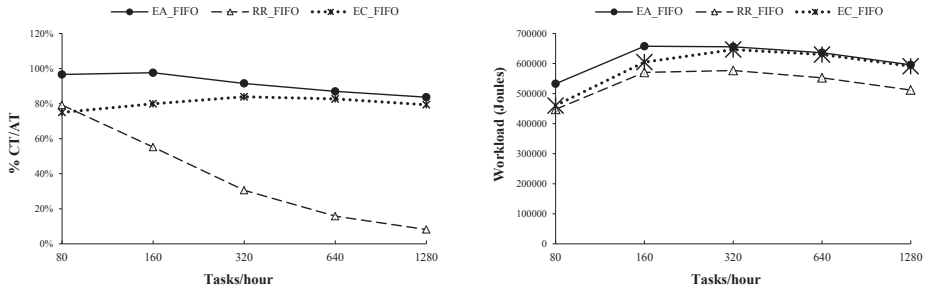


Fig. 3.37. (a) Ratio of completed tasks (CT) and assigned tasks (AT) and (b) Workload w.r.t. task arrival rate using EA, RR and EC with FIFO policy.

are always higher than those of EA and RR but for frequency values greater than 320 the difference gets smaller. For the FIFO case all the strategies take higher times: RR is faster than EA only for frequency values greater than 320. However, the difference is important only for very high frequency values. On the contrary, EC is always significantly slower than both EA and RR.

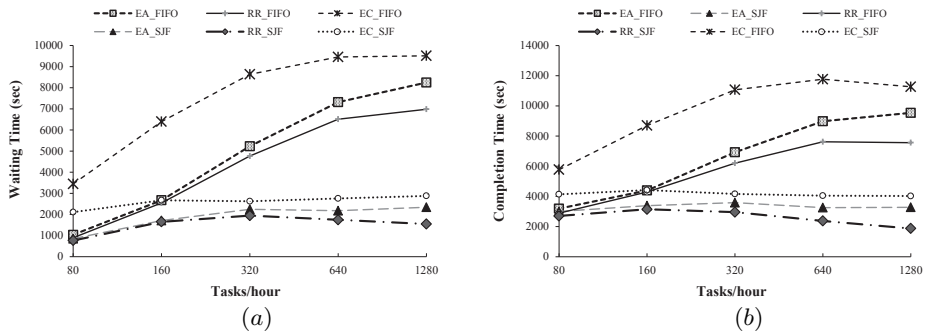


Fig. 3.38. (a) Waiting Time and (b) Completion Time w.r.t. task arrival rate using EA, RR and EC with FIFO and SJF policies.

The aim of this fourth set of experiments is to evaluate the performance of EA against RR and EC by varying the distribution of the initial energy level among the devices in the network. In particular, we considered four energy configurations:

- CONF_1: the energy is assigned to the devices following a normal distribution with average at 40% of the peak value (this configuration has been used also for the experiments presented above).
- CONF_2: all the devices start with the same energy level, equal to 40% of the peak value.

- CONF_3: the energy is uniformly distributed in the range 15%-65% of the peak value.
- CONF_4: the energy is assigned using a bimodal distribution, with local maxima at 30% and 75% of the peak value.

For all the configurations, the total energy level available is constant and equal to about 700 kJoule. In each configuration, 2500 tasks are submitted with a frequency $\lambda = 80$ tasks per hour.

Figures 3.39(a), 3.39(b), and 3.40 show only the results for the FIFO policy because they are similar to those of SJF. EA and EC adapt their behavior to the different configurations always assigning the tasks respectively to the most energy powerful devices and to the less energy powerful ones. Differently, RR allocates the tasks without taking into account the energy availability of the devices. Thus, the performance of RR depends on the specific combination of devices and tasks: each time a task has to be allocated, the effectiveness of RR depends on the remaining energy of the device at the beginning of the scheduling queue and on the load of the task to be allocated. For example, if the most charged device is at the beginning of the scheduling queue when a high computational task has to be allocated, there is a chance of executing the task without turning off the device. This is particularly evident in Figure 3.39(a) for configurations CONF_1 and CONF_3 in which 90% of devices is turned off. Figure 3.39(a) shows that EA keeps alive almost the same number of devices in the different network configurations, and this number is always greater than the number of alive devices of RR. EC keeps alive a variable number of devices and this number is always lower than that of EA. This is because EC consolidates the devices by assigning the tasks to the less energy powerful devices, causing this way the turning off of the weakest devices. For this reason, EC remains with a lower number of alive devices that are more loaded compared to the ones of EA. Thanks to this, EC is able to execute heavier tasks, and, thus, this is the reason for the difference in the number of executed tasks with EA as confirmed by Figure 3.40, which shows that the workload carried out by both strategies is the same.

3.6 Conclusion

Distributed computing over mobile ad hoc networks (MANETs) demands for effective scheduling strategies addressing both the energy constraints of battery-operated wireless devices and the decentralized nature of MANETs. The Energy-Aware (EA) scheduling strategy proposed here was designed to address this two-fold need. To maximize network lifetime and the number of alive devices, the EA scheduler implements a heuristic algorithm that balances the energy load among all the devices exploiting a cluster-based architecture. An experimental evaluation has been performed to assess the performance of the EA strategy in different network and application scenarios. The experimental results showed that by using the proposed energy-aware task allocation

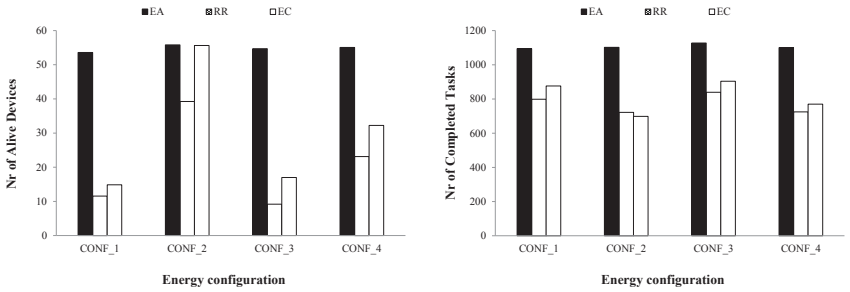


Fig. 3.39. (a) Number of alive devices and (b) Number of completed tasks w.r.t. energy configurations, using EA, RR and EC.

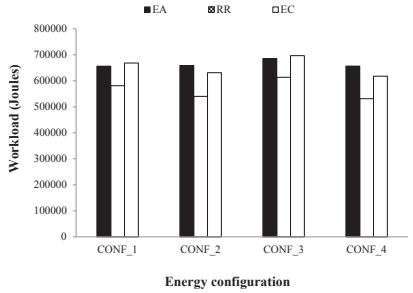


Fig. 3.40. Workload w.r.t. energy configurations, using EA, RR and EC.

approach, the network lifetime can be extended and the number of alive devices can be significantly higher compared to related scheduling strategies, while meeting application-level performance constraints. Based on these results, we can conclude that the EA approach combined with a cluster-based architecture is an effective strategy to support distributed task execution in mobile computing scenarios.

4

Analysis of Mobile Social Data

Social networks are becoming one of the most popular media for sharing and communicating among individuals worldwide. As more people start to use social networks and post and tweet status data, more information about the personal lives of individuals begins to leak out into cyberspace.
Social media is receiving an always increasing attention from the research community as many of devices combine the features of mobile phones with GPS navigation units, embedding precious information concerning human dynamics and behaviors within urban context. The ability to associate spatial context to social posts in on line social network is becoming a popular feature of the most used on line social networks. Facebook and Twitter exploit the GPS readings of users phones to tag user posts, photos and videos with geographical coordinates. So the number of tweets with geographic information is progressively increasing. According to this view, people move in the real world visiting a sequence of locations and generating this way many trajectories in the form of geo-tagged tweet sequences. When a person sends a tweet from a geographic location, that tweet becomes a digital footprint marking her/his physical presence [81].

In this thesis, we leverage the power of social networks and geo-tagged social data to analyze users spatial-temporal movements detected through their interaction with social networks, to discover people and community behavior, i.e. patterns, rules and regularities in moving trajectories. The basic assumption is that people often tend to follow common routes: e.g., they go to work every day traveling the same roads. Thus, if we have enough data to model typical behaviors, such knowledge can be used to predict and manage future movements of people.

To this end, we defined and implemented a methodology to mine popular travel routes from geo-tagged posts of an urban area such as those collected from Twitter. The proposed methodology consists of various phases. The first step of the methodology is the *collection of geo-tagged data* from Twitter. Due to the variable accuracy of GPS, it is necessary to *cluster the locations* of the geo-tagged tweets so that each place is identified by a single pair of geographic

coordinates and associate to them a semantic label exploiting Foursquare (*semantic location detection*). The result of this step depends on the extension of the geographical area in the city we are interested in, ranging from a specific venue (e.g., school, bar, home, workplace) to an area (e.g., Trafalgar Square in London) or a suburb (e.g., Covent Garden in London) or a district (e.g., zone 1 in London). After locations detection we *generate trajectories or travel routes*. We can then *mine frequent travel routes* using sequential pattern mining. If a trajectory pattern occurs frequently, we consider it as a *frequent travel route*. Finally, we extract spatial-temporal information for each of them to capture the factors that may drive users' movements. In particular, for all the frequent patterns, we compute a set of daily snapshots concerning the visited places, the movements among them, and the duration of the visits at each location. As result, we provide the top interesting locations and frequent travel sequences among these locations, in a given geo-spatial region. For interesting locations we mean the culturally important places, such as National Gallery or Buckingham Palace in London (i.e. popular tourist destinations), and commonly frequented public areas, such as shopping malls/streets, restaurants, cinemas and bars. With the information mentioned above, individuals can understand an unfamiliar city in a very short period and plan their journeys with minimal effort.

The rest of the chapter is organized as follows. Section 4.1 presents a description of on line social networks and overviews related works. Section 4.2 presents the methodology proposed, and Section 4.4 emphasizes the complex *semantic location detection* phase presenting a *Category Classifier* to infer places category. In Section 4.3 we describe the statistical analysis of frequent travel routes; we first introduce a set of features characterizing the travel routes (4.3.1) and, then, describe the analysis performed on the collected geo-tagged tweets (4.3.2). Finally, Section 4.5 concludes the chapter.

4.1 Background and Related Work

This chapter introduces and discusses on-line social networks and location-based social networks, briefly presenting for both of them the most popular ones (some of them are shown in Figure 4.1[1]). At a later stage, new challenges and opportunities for data analysis are presented.

4.1.1 On-line Social Network

The potential of computer networking to facilitate newly improved ways of computer-mediated social interaction was suggested early on [11]. As the Internet grew towards the end of the 20th century, many social networking services and concepts began to arise. A social network is a social structure made up of

[1] blog.socialmaximizer.com

Fig. 4.1. Most popular on-line social networks and location-based social networks.

individuals connected by one or more specific types of interdependency, such as friendship, common interests, and shared knowledge. Generally, a social networking service builds on and reflects the real-life social networks among people through online platforms such as a website, providing ways for users to share ideas, activities, events, and interests over the Internet. Services such as *America Online* and *Geocities* allowed users to share personal information in the form of easy to use webpages and communication mediums. In 2002 *Friendster* was launched, heralding a new age of online social networking. A year later, *MySpace* was launched. MySpace is a place where a user can create a profile page that (s)he can use to meet new friends. However, the popularity of MySpace would be short lived. In 2004, *Facebook* launched and today, is still the most popular online social network[2]. While Facebook dominates the market, there are many other popular online social networking websites in use today such as *Twitter*, *LinkedIn*, *Pinterest*, *Google Plus+*. These online social networking websites have grown in popularity over the years. Millions of users post status updates and share personal information on these websites. Most social networking websites provide developers with an Application Programming Interface (API) to interact with and gather information from the social networking services. Today, entire companies are formed around using information from and monetizing the users of these online social networking websites. With so many updates posted to these websites, there is much information and knowledge to be gained about human social interaction.

Facebook

Facebook[3] is a popular free social networking website that allows registered users to create profiles, upload photos and video, send messages and keep in

[2] www.ebizmba.com
[3] www.facebook.com

touch with friends, family and colleagues. Users on Facebook tend to share more personal information due to the tightly knit friendship nature of the site. Before Facebook users are allowed to communicate with each other and view status updates, a mutual friendship between the users is required. Facebook offers a range of privacy options to its members. A member can make all his communications visible to everyone, he can block specific connections or he can keep all his communications private. Members can choose whether or not to be searchable, decide which parts of their profile are public, decide what not to put in their news feed and determine exactly who can see their posts. For those members who wish to use Facebook to communicate privately, there is a message feature, which closely resembles email.
The site, which is available in 37 different languages, includes public features such as:

- *Marketplace* - allows members to post, read and respond to classified ads.
- *Groups* - allows members who have common interests to find each other and interact.
- *Events* - allows members to publicize an event, invite guests and track who plans to attend.
- *Pages* - allows members to create and promote a public page built around a specific topic.
- *Presence technology* - allows members to see which contacts are online and chat.

Within each member's personal profile, there are several key networking components. The most popular is arguably the *Wall*, which is essentially a virtual bulletin board. Messages left on a member's Wall can be text, video or photos. Another popular component is the *virtual Photo Album*. Photos can be uploaded from the desktop or directly from a smartphone camera. There is no limitation on quantity, but Facebook staff will remove inappropriate or copyrighted images. An interactive album feature allows the member's contacts (who are called generically called "friends") to comment on each other's photos and identify (tag) people in the photos. Another popular profile component is status updates, a microblogging feature that allows members to broadcast short Twitter-like announcements to their friends. All interactions are published in a news feed, which is distributed in real-time to the member's friends.

In May 2007, Facebook opened up its developers' platform to allow third-party developers to build applications and widgets that, once approved, could be distributed through the Facebook community. In May 2008, Facebook engineers announced Facebook Connect, a cross-site initiative that allows users to publish interactions on third-party partner sites in their Facebook news feed[4].

[4] whatis.techtarget.com

Twitter

Twitter[5] is an online social networking service that enables users to send and read short 140-character messages called *tweets*.

Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone and Noah Glass and by July 2006 the site was launched. The service rapidly gained worldwide popularity, with more than 100 million users who in 2012 posted 340 million tweets per day[6]. Twitter has emerged as a new medium in spotlight, in fact its usage spikes during prominent events. For example, a record was set during the 2010 FIFA World Cup or when American singer Michael Jackson died on June 25, 2009.

It's a platform wherein users share their thoughts, news, information and jokes. Twitter makes global communication cheap and measurable. Profiles are (usually) public unless a user elects to make his profile private.
Twitter users follow others or are followed. Unlike on most online social networking sites, such as Facebook or MySpace, the relationship of following and being followed requires no reciprocation. A user can follow any other user, and the user being followed need not follow back. Being a follower on Twitter means that the user receives all the tweets from those the user follows.
Common practice of responding to a tweet has evolved into well-defined markup culture: *RT* stands for retweet, @ followed by a user identifier address the user, and # followed by a word represents a *hashtag*. This well-defined markup vocabulary combined with a strict limit of 140 characters per posting conveniences users with brevity in expression. The retweet mechanism empowers users to spread information of their choice beyond the reach of the original tweets followers [60].

Twitter places great reliance on open-source software. In the early days of Twitter, tweets were stored in MySQL databases that were temporally sharded, that is the databases were split by time of posting. MySQL was causing problems with both reading and writing to Twitter and the company decided to switch to a Java server they call Blender. The service's application programming interface (API) allows other web services and applications to integrate with Twitter.

LinkedIn

LinkedIn[7] is a business-oriented social networking service. LinkedIn's founders are Reid Hoffman, Allen Blue, Konstantin Guericke, Eric Ly and Jean-Luc Vaillant. The site was officially launched on May 5, 2003. The company is publicly held and has a diversified business model with revenues coming from

[5] www.twitter.com
[6] blog.twitter.com
[7] www.linkedin.com

talent solutions, marketing solutions and premium subscription products. It is currently available in 23 languages[8].

The basic functionality of LinkedIn allows users (workers and employers) to create profiles and "connections" to each other in an online social network which may represent real-world professional relationships. Users can invite anyone (whether a site user or not) to become a connection. However, if the invitee selects "I don't know" or "Spam", this counts against the inviter. If the inviter gets too many of such responses, the account may be restricted or closed.
This list of connections can then be used in a number of ways:

- Obtaining introductions to the connections of connections (termed second-degree connections) and connections of second-degree connections (termed third-degree connections).
- Users can find jobs, people and business opportunities recommended by someone in one's contact network.
- Employers can list jobs and search for potential candidates.
- Job seekers can review the profile of hiring managers and discover which of their existing contacts can introduce them.
- Users can post their own photos and view photos of others to aid in identification.
- Users can follow different companies and can receive notifications about the new joining and offers available.
- Users can save (i.e. bookmark) jobs that they would like to apply for.
- Users can "like" and "congratulate" each other's updates and new employments.
- Users can see who has visited their profile page.

4.1.2 Location-Based Social Network

To gain more users and maintain their current user base, online social networking websites must constantly revise and add new features to their services to keep the content fresh and exciting for the users. One of the popular features being added to these services is the idea of attaching geo-location information to status updates. The increasing availability of location-acquisition technology (for example GPS and Wi-Fi) empowers people to add a location dimension to existing online social networks in a variety of ways. *Foursquare* was one of the original websites to attach such geo-location information to status updates. Users can upload location-tagged photos to a social networking service such as *Flickr*, record travel routes with GPS trajectories to share travel experiences in an online community, for example *GeoLife*, or log jogging and bicycle trails for sports analysis and experience sharing, as in *Bikely* [12]. These kinds of location-embedded and location-driven social structures are

[8] press.linkedin.com

known as *location-based social networks* (LBSN). Here, a location can be represented in different way:

- absolute, i.e., latitude-longitude coordinates;
- relative, e.g., 100 meters north of the Space Needle);
- symbolic, e.g., home, office, or shopping mall.

The ability to associate spatial context to social posts is becoming a popular feature of the most used on line social networks. For instance, Facebook and Twitter have started allowing users to tag their posts with geographical coordinates collected through the GPS interface of users smart phones.

User and location are two major subjects closely associated with each other in a location-based social network. As illustrated in Figure 4.2[9], users visit some locations in the physical world, leaving their location histories and generating location-tagged media content. If we sequentially connect these locations in terms of time, a trajectory will be formulated for each user [12].

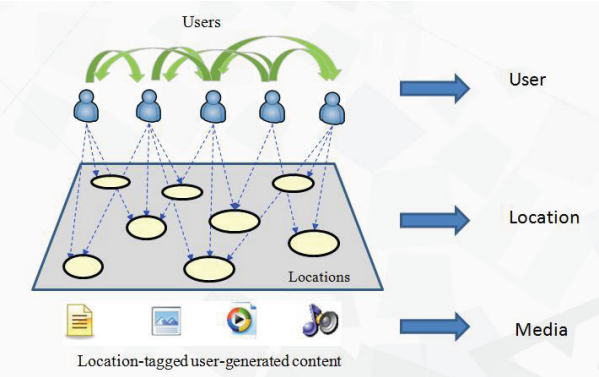


Fig. 4.2. Structure of a location based social network (LBSN).

Foursquare

Foursquare[10] is a local search and discovery service mobile app, it has clients for smartphones such as iPhone, BlackBerry, Palm, and the Android platform. The service was created in late 2008 and launched in 2009 by Dennis Crowley and Naveen Selvadurai. Foursquare lets people connect to friends, which are equivalent to the concept of friends on other online social networks, but it is a rather unique social networking site because the main focus of status updates are locations in the form of *check-ins* to venues in the area. Users check in to a location by physically going there and using their phones GPS to send a

[9] lms.comp.nus.edu.sg/content/area-4-location-analytics
[10] www.foursquare.org

status update. When doing a check-in, Foursquare examines the users current location and shows a list of nearby places. Users can also register new places. When a user checks in to a place, a check-in notification is by default pushed to their Foursquare contacts. People can choose to be notified of all check-ins by their contacts. At the time of the check-in, users can also decide if they want to check-in off-the-grid, in which the check-in is recorded by Foursquare but not shared with contacts.
People can also connect their Foursquare account to other online services, such as Facebook and Twitter, and have their check-ins be announced on these services. Users who have checked-in to a place can also see who else has recently checked-in (Whos here). Users can also allow local businesses to view checkins to their location. The game aspect of Foursquare offers virtual and tangible rewards for check-ins. Virtual rewards come in the forms of points, badges, and mayorships visible in ones public profile. Badges are awarded for a variety of reasons, e.g. for starting to use the service, checking-in on a boat, checkingin with 50 people at the same time, or checking-in at a special event. Mayorships are awarded to a single individual for having the most check-ins in a given place in the past 60 days, where only one check-in per day is counted. Some companies offer discounts for mayors of a place, for example, some coffee shops offer discounts on coffee. Foursquare also enables social recommendations through tips, a small snippet of text associated with a place. Tips are intended to suggest possible activities for that place [61].

The requirement of attaching a physical location to check ins makes Foursquare a very interesting social network to data mine. Using Foursquare data from a user, it is easily possible to trace a general path that the user took over time. This ability to trace user activity is very useful for creating location aware applications that work together with Foursquare. Foursquare aims to provide highly personalised recommendations of the best places to go around a user's current location.

The requirement to attach location to Foursquare updates leads to a multitude of privacy concerns. Due to the fact that much of the user data on Foursquare is considered sensitive, the Foursquare API relating to users is very restrictive. Each individual must authenticate themselves with the application before data collection is allowed.

Flickr

Flickr[11] is a photo-sharing website where anyone can upload and tag photos, browse others' photos, and add comments and annotations. Users can create photo sets and collections to manage content, and participate in topical groups to cultivate a sense of community. Launched in February 2004, Flickr embodies what has come to be known as Web 2.0 technology. The site provides the tools, but the value derives from the contributions of the user community photos,

[11] www.flickr.com

comments, ratings, and organization and connections that the site facilitates between individuals. Flickr also provides a range of privacy settings, giving users considerable control over how their photos can be used.

The site has more than two billion images and 20 million unique tags, and its collaborative tools have made it popular in higher education. Some faculty have begun using Flickr images in their courses, and art schools, biologists, and others use the site to share, critique, and analyze visual information.

Flickr offer *free accounts*, which have limits on bandwidth and the number of groups in which each photo can be included, and *paid accounts*. Using any of several tools or an online form, registered users can upload photos to the site, assign tags, indicate whether each photo is public or private, and select other settings, including copyright, which can be traditional all rights reserved or any of the Creative Commons licenses. Copyright settings can vary for different photos. Users can also specify who can add comments, tags, or annotations to their photos, and each photo can also include information about how and where it was taken. Photos can be organized by groups, which also serve as community topics. A group might be called wooden canoes, and photos included in that group (from multiple contributors) will appear on its page along with a list of the groups members and a discussion board, for which an RSS feed is available. Flickr photos integrate nicely with blogs, and the site offers an API, which allows developers to write other applications that can interact with Flickr content. For each photo, the contributors name links to that users profile page. Tags and group names link to dynamic collections of all Flickr material similarly identified. Profile pages include details provided by the contributor, as well as testimonials from other users and an e-mail tool to send private messages to that contributor. The result is an always-changing network of collections and connections, fostering a strong sense of community among users[12].

Instagram

Instagram[13] is an online mobile social networking service that enables its users to take pictures and videos, and share them on a variety of social networking platforms, such as Facebook, Twitter, Tumblr and Flickr. It allows to experience moments in users friends' lives through pictures as they happen. Instagram was created by Kevin Systrom and Mike Krieger, and launched in October 2010.

Today, Instagram isnt just used to post photos. Its becoming an assist for brands to reach more visitors (thanks to sponsored photos and videos), or researchers to discover the happiest city in the UK. Between being able to share Instagram images on your Facebook, Twitter, Tumblr and Flickr accounts, and the implementation of Instagram Direct (which allows users to

[12] net.educause.edu

[13] www.instagram.com

send photos only to a specific user or group of users, its a safe bet that the photo-sharing, video-sharing and social networking site is only going to get bigger and stronger as time goes on[14].

On Instagram, a user can follow other users' photo streams as they post them and he can be followed back by those users (or other users) as well. Also, it is possible to search for friends by name or find friends that are already connected to a user on other social networks like Facebook or Twitter.
When a user take photos using the Instagram app, they will always appear in the feeds of the users who are following him. He can browse through the *Explore tab* to find new users to follow and creative photos to look at.
Instagram offers a number of photographic *filters* that users can apply to their images[15]:

- *Normal*: No filter applied.
- *1977*: The increased exposure with a red tint gives the photograph a rosy, brighter, faded look.
- *Amaro*: Adds light to an image, with the focus on the centre.
- *Brannan*: Increases contrast and exposure and adds a metallic tint.
- *Earlybird*: Gives photographs an older look with a sepia tint and warm temperature.
- *Hefe*: Hight contrast and saturation, with a similar effect to Lo-Fi but not quite as dramatic.
- *Hudson*: Creates an "icy" illusion with heightened shadows, cool tint and dodged center.
- *Inkwell*: Direct shift to black and white - no extra editing.
- *Kelvin*: Increases saturation and temperature to give it a radiant "glow".
- *Lo-fi*: Enriches color and adds strong shadows through the use of saturation and "warming" the temperature.
- *Mayfair*: Applies a warm pink tone, subtle vignetting to brighten the photograph center and a thin black border.
- *Nashville*: Warms the temperature, lowers contrast and increases exposure to give a light "pink" tint - making it feel "nostalgic".
- *Rise*: Adds a "glow" to the image, with softer lighting of the subject.
- *Sierra*: Gives a faded, softer look.
- *Sutro*: Burns photo edges, increases highlights and shadows dramatically with a focus on purple and brown colors.
- *Toaster*: Ages the image by "burning" the centre and adds a dramatic vignette.
- *Valencia*: Fades the image by increasing exposure and warming the colors, to give it an antique feel
- *Walden*: Increases exposure and adds a yellow tint.
- *Willow*: A monochromatic filter with subtle purple tones and a translucent white border.

[14] www.searchenginejournal.com
[15] webtrends.about.com

- *X-Pro II*: Increases color vibrance with a golden tint, high contrast and slight vignette added to the edges.
- Recently Instagram has added a new feature when editing photographs. The button with a picture of a sun is known as the *Lux* effect. This tool allows you to quickly adjust the exposure and contrast through a simple 100-point slider. This new version of editing allows you to control the brightness to the saturation levels of each photograph.

4.1.3 Social Data Analysis and Research Issues

The geo-tagged data, in some on-line social networks (e.g., Facebook and Twitter) and LBSNs, contain spatial-temporal social context and present new challenges and opportunities for data analysis. In the following we present a lot of studies on human mobility and its implications in the social relationships and location-based services.

Several studies have been focused on the *connection between social tie and geographical distance.*

Scellato et al. [64] proposed two geo-social metrics, embedding the geographical distance into social structure, to measure the node locality and geographical clustering coefficient. Two findings are presented in this work: 1) users who live close have higher probability to create friendship links than those who live distant; and 2) users in the same social cluster show short geographical distances. Furthermore, the authors compared location-based social networks (Brightkite[16] and Foursquare) with content sharing based social networks (LiveJournal[17] and Twitter), discovering the difference of network properties between these two kinds of social networks. They found that people within a social cluster on the LBSNs tend to have smaller geographical distance than those online social networks focusing on content producting and sharing.

Cranshaw et al. [82] examines the location traces of Facebook for discovering relationships between the users mobility patterns and structural properties of their underlying social network. They introduce a set of location-based features for analyzing the social context of a geographic region, including location entropy, which measures the diversity of unique visitors of a location. Using these features, they provide a model for predicting friendship between two users by analyzing their location trails. They have also shown that the entropy of the locations a user visits can provide insight into the number of ties that the individual has in the social network.

Volkovich et al. [83] study the interplay between spatial distance, interaction strength and structural properties in social ties arising among members of an online social platform, Tuenti. They discover that spatial constraints on online social networks are intimately connected to structural network properties,

[16] brightkite.com
[17] www.livejournal.com

with important consequences for information diffusion.
Brown et al. [65] extended the research on LBSNs to social community, and discovered that the rise of social groups is affected by both social and spatial factors. They reported that social communities on location-based social networks seem to be more relevant to the spatial factor.

Other approaches have been focused on the ***Semantic place labeling***. It has become a popular research direction given the importance of associating context to geographical coordinates. In this work, we have addressed this issue, because while geographical information is quite useful and already indicative of user behavior, it also lacks some semantics about the type of place the user is (e.g., restaurant, museum, school) which would allow a better understanding of users' patterns. Various approaches for semantic place labeling have been proposed recently.
In the work by Liao et al. [118] the authors used features including the time of visits and the presence of bus stops, restaurants, and other points of interests to automatically label places. The main innovation is the introduction of a hierarchical conditional random field (CRF) that aids inference accuracy by exploiting the temporal sequence of place visits, e.g., works often follows home. Their system was tested on GPS trajectories of only four people.
Chen et al. [95] followed a similar approach but they processed label sequences with a hidden Markov model rather than a CRF. Our approach attempts to automatically label places based on a mapping between features describing a visit to a place and the place label.
In [96] the authors developed a classifier called Placer that classifies locations into different label categories based on the timing of visits to that place, nearby businesses, and simple demographics of the user. Our work shares the same aim as [96] of inferring location category by exploiting the timing visits to a place. However, this is the only common feature. We aim to classify locations by exploiting the tweets whereas [96] uses as data sources two different diary surveys conducted in the U.S and only one of which includes latitude/longitude data. Another important difference between the two approaches is that while in our case we have to identify and extract features by analyzing the tweets, in [96] the statistics used as features are already available from the surveys not requiring, thus, any analysis and processing of data as it is instead necessary in our case. Furthermore, the features used in [96] are different from the ones we used. In fact, [96] uses demographic data like the age and the gender of users together with nearby business and point of interests (POI) as features for the classifier.

Another traditional task in mobile computing is *Location prediction*. It has been studied over a long period. Researchers analyze human mobility patterns to improve location prediction services, and therefore exploit their potential power on various applications such as mobile marketing [119, 120], and traffic planning [70, 121]. Current research on location prediction in LBSNs mainly focuses on two tasks: 1) predicting a users home location; and 2) predicting a users location at any time. The former task considers the static home location

of a user, while the latter considers more about a users moving trajectories, with his location in movement [13].

The motivation of *home location prediction* arises from the sparsity of available user home locations on popular social networks such as Twitter and Facebook [71]. Current work in home location prediction have been focused on the relationship between the geographic context and the user-generated content [66, 67].

Authors in [66] present a method for identifying geographically-aligned lexical variation directly from raw text. Their approach is a multi-level generative model that reasons jointly about latent topics and geographical regions. The resulting system has multiple capabilities, including: (i) analyzing lexical variation by both topic and geography; (ii) segmenting geographical space into coherent linguistic communities; (iii) predicting author location based on text alone.

Differing from [66] that aims to estimate a single location for each user, the goal in [67] is to infer the location of each new tweet, based on its content and the author's other tweets. They describe a joint hierarchical model of location and content, both personalized to the individual preferences of a user. More precisely, they build a hidden tree structure for topics occurring in tweets, assuming that these topics are location specific. Thus, they adorn this tree with locations at which such topics could be observed, and represent users by modeling distributions over leaves within the overall tree. They perform posterior inference and apply the framework to organize tweets into a hierarchical structure and show that this tree structure can be used to better predict locations of unlabeled tweets.

To predict a users location at any time, usually referred to as *next location prediction*, researchers start to investigate the role of social friendship in explaining a users mobile patterns [72]. On the other hand, leveraging social networking information for location prediction also becomes a new challenge, since how to embed the social property into geographical patterns is a still an open issue on location based social networks. Current work on LBSNs has proposed various approaches to combing social network information with traditional spatial-temporal patterns [13].

Chang et al. [73] utilized logistic regression model to combine a set of features extracted from Facebook data. The features include a users previous check-ins, users friends check-ins, demographic data, distance of place to users usual location, etc. Their results demonstrated that the number of previous check-ins by the user is a strong predictor, and also previous check-ins made by friends and the age of the user are good features for prediction.

Linear combination has been mostly used for integrating social friendship with spatio-temporal patterns. Cho et al. [62] considered the user check-in probability as a linear combination of social effect and non-social effect. The social effect assumes the check-in of a user to be close to the check-ins of his friends, both in space and in time; while the non-social effect captures the periodical patterns, which considers the users personal movement following a 2-D Gaus-

sian distribution, with the two Gaussian centers focusing on home and work. Noulas et al. [86] predict the next place in a city that the user will visit, proposing a set of features that exploit information on transitions between types of places, mobility flows between venues, and spatio-temporal characteristics of user check-in patterns. They combining all features in two supervised learning models, based on linear regression and M5 model trees, resulting in a higher overall prediction accuracy.

A variety of papers has been centered on characterizing *human mobility in urban areas* to study and delineate crowd mobility [84, 68], or to perform world landmark localization [85].

Cheng et al. [74] explored millions of check-ins on Facebook, and observed various spatial, temporal and social patterns. For example, people tend to move to nearby places and occasionally to distant places. The authors observed that user mobility is influenced by social status, geographical and economic factors. Furthermore, the user check-in behavior presents strong daily/weekly patterns and periodic property, indicating the potential to improve location-based applications.

In [75], the authors observed similar geo-temporal patterns of check-ins on weekdays and weekends. They reported that around 20% of consecutive check-ins in Foursquare happen within 1 km of one another, 60% between 1 and 10 km, and 20% over 10 km.

Li et al. [76] studied users mobility characteristics on Brightkite. They clustered users based on their mobility patterns derived from user updates and movement paths, and obtained four user groups, namely home users, home-vacation users, home-work users, and other users which present different mobility patterns from previous groups.

Although the literature presents a large variety of studies on social networks, to the best of our knowledge, few previous studies have been conducted on the extraction of trajectories by exploiting the geo-tagged social posts.

Trajectory pattern mining is the challenge that we faced in this work. By way of explanation, the time and geo-coordinates associated with a sequence of mobile social data manifest the spatial-temporal movements of people in real life. We analyze such movements to discover people and community behavior. To this end, we defined and implemented a methodology to mine popular travel routes from geo-tagged posts and, then, we identified a set of features characterizing such travel routes. Our approach infers interesting locations and frequent travel sequences among these locations in a given geo-spatial region.

Our work shares the same aim as [87] of providing the most frequent travel routes and the top interesting locations in a given geo-spatial region. However, the authors in [87] mine trajectory patterns from the spatial distribution of millions of photos uploaded on Flickr. They propose a ranking mechanism to obtain more informative trajectory patterns. To diversify the ranking results they adopt longest common sub-sequence(LCSS) as the similarity measure and use an exemplar-based algorithm. They associate semantics to the loca-

tions on the basis of associated tags that are contributed by users for each photo. In contrast, the geo-tagged tweets are lacking of such information, due to this we perform the semantic location detection by queering a Foursquare dataset. Moreover, we deal a huge quantity of data, mining trajectory patterns characterized by a very high number of users visiting the sequence of locations. In particular their most frequent trajectory ($LondonEye \longrightarrow BigBen$) is traveled by 21 users and this is the maximum value that they get, in our case the same trajectory is traveled by 176 users. After the ranking they obtain trajectories with a longer travel sequence, but their result decline in terms of number of users, in fact their top ranked trajectories patterns are traveled at most by 2 or 3 users.

Other proposals that aim to extract patterns from photos with textual and spatio-temporal metadata are [77, 78, 94]. In particular [94] explored the construction of travel itineraries from geo-tagged photos. Using also Flickr, the authors have used a pool of geo-tagged photos and its semantic content to investigate the tourist traffic flow among different locations by exploiting Markov chain model, and the topological characteristics of travel routes. Their focal point is analyze tourist mobility, while we also consider residential travel routes. Moreover, they not extracted temporal features such as the mean duration of a journey or other timing statistics which we define (e.g., time of day and day of week of the typical journey).

A different strand of research examines the mobility patterns in a city during a certain event by observing microblog posts. As an example [89, 90] used Twitter as a sensor to detect natural phenomena. Twitter has also been the focus of the work by [88] which used a dataset of geo-tagged tweets to analyze the significant alteration of the beat of Barcelona when a city-wide event happens. As in our approach, they use Foursquare to associate semantic to the locations, and like us, they show the category of place mostly traversed in the travel routes. The common point with our work is to understand the mobility pattern in a city, but the main difference is that we extract from the geo-tagged tweets the sequence of locations along with the users move considering a set of random places, while they consider a fixed set of regions, i.e. the districts included in the city, and analyze the trajectories traversing among them. They build OD matrices and perform an analysis more coarse grain compared with ours, showing anomalies within people displacements during the event.

A lot of research work has be done in trajectory pattern mining [79, 80, 91, 92]. In particular, [91] describes movement patterns in both spatial and temporal contexts, based on RoI (Region of Interest) and TAS (Temporally Annotated Sequences), which are an extension of sequential patterns with transition time between points. Also with respect to this set of works, our approach presents elements of novelty. Fist, we deal with the more unpredictable and irregular data of the Twitter social network, whereas they use GPS traces of mobile devices or synthetic datasets for the test. In addition, they lack some semantics

about the type of place in the travel sequences, which would allow a better understanding of users' patterns.

4.2 Methodology

In this section we describe how to mine popular travel routes from social geo-tagged data such as those collected from Twitter. Figure 4.3 outlines the main steps of the methodology.
The first step is the *collection of geo-tagged data*; in our case we used Twitter as data source, from it we extracted tweets tagged with a GPS location.
The second step is *semantic location detection*. Locations are detected through dense clustering that allows to cluster GPS coordinates into specific places and associate them to Foursquare categories when available. The extension of the geographical area in a city we are interested in, ranges from a specific venue (e.g, school, bar, home, workplace) to an area (e.g., Trafalquasre in London) or a suburb (e.g., Covent Garden in London) or a district (zone 1 in London). The granularity of the geographical extension can be set through appropriate tuning of clustering parameters, as will be detailed in the following.
After locations detection we *generate trajectories or travel routes*. Obviously, the granularity of travel routes depends on the granularity of the above clustering step. Therefore, we could obtain travel routes among venues in the city or travel routes among areas in the city or among suburbs. The next phase consist in *mine frequent travel routes* using a sequential pattern mining algorithm. If a trajectory pattern appears often, we consider it as a *frequent travel route*. Finally, we extract *spatial-temporal features* so as to capture the factors that may drive users' movements. In particular, for all the frequent patterns, we compute a set of daily snapshots such as the visited places, the movements among them, the duration of the visits at each location.
In the following of the section we will describe in detail each of the steps of the proposed methodology.

4.2.1 Twitter dataset and data model

The geo-located data mined in this work is a dataset of tweets tagged with GPS location within the boundaries of the city of London, one of the top three cities by number of tweets. Twitter provides a large quantity of data due to the public nature of the social networking site. Since by default all profiles are set to public, Twitter lends itself well to anonymous data mining. Twitter also does not require any user authentication to use their API. Since Twitter updates are geo-coded with the location of the user at the time of the status update, Twitter is an ideal resource for analyzing human mobility. Numerically speaking, we consider a Twitter dataset of 7,424,112 tweets issued by 292,195 mobile users in 6,098,148 distinct locations, during a period of six month from June 2013 to November 2013.

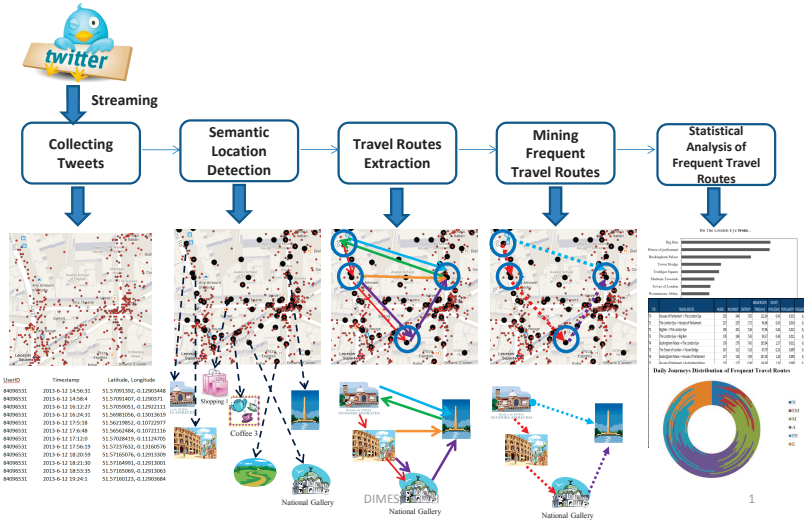


Fig. 4.3. Methodology.

We formulate the problem in terms of a specific social network, Twitter, and its geo-localized tweets, although the general formulation is applicable to any other social network with geo-tagged posts. Given a set of geo-tagged tweets TW, we extract the set of locations L visited from Twitter users U.

Definition 4.1. ***Geo-tagged tweet.*** *A geo-tagged tweet* $\mathrm{tw} \in \mathrm{TW}$ *can be expressed as a triple in the form:*

$$\mathrm{tw} =< \mathrm{u}, \mathrm{l}, \mathrm{t} >.$$

It is characterized by a user u *who created the tweet, a location* l *from where* tw *has been posted, and a timestamp* t *that is the time at which it has been posted.*

Definition 4.2. ***Location.*** *A location* $\mathrm{l} \in \mathrm{L}$ *is represented by a triple:*

$$\mathrm{l} =< (\mathrm{lat}, \mathrm{lon}), [\mathrm{p_l}], 4\mathrm{sq_l} >$$

where: $(\mathrm{lat}, \mathrm{lon})$ *is the pair of geographic coordinates that identify the location;* $[\mathrm{p_l}]$ *is an array containing its properties; and if exist,* $4\mathrm{sq_l}$ *is the category of the most-likely Foursquare venue that can be associated to* l*. The properties vector* $[\mathrm{p_l}]$ *of a location* l *contains:* $\mathrm{U_l}$*, the set of people who visited* l*;* $\mathrm{TW_l}$*, the set of tweets posted in* l*;* $\mathrm{V_l}$*, the set of the visits of* l*;* $\mathrm{tIn_l}$*, the total time that users spend in* l*; and* $\mathrm{nD_l}$*, the total number of days in which* l *is visited.*

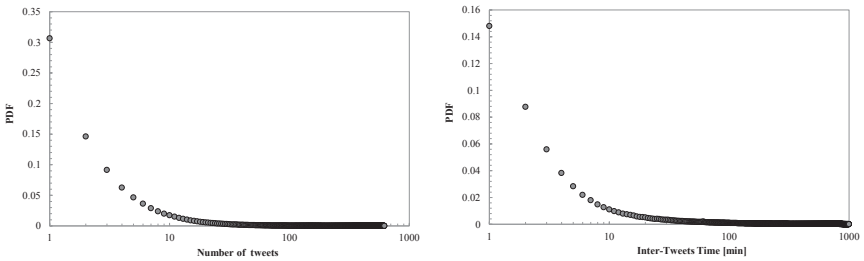
Definition 4.3. ***Visit.*** *A visit* v *is expressed as a quadruple:*

$$\mathrm{v} =< \mathrm{u}, \mathrm{l}, (\mathrm{tw_{first}}, \ldots, \mathrm{tw_{last}}), \Delta\mathrm{v} >$$

It is characterized by: u*, the user who visits the location* l*;* $(\mathrm{tw}_{\mathrm{first}}, \ldots, \mathrm{tw}_{\mathrm{last}})$*, a sequence of tweets that* u *posts in* l *before moving to another place; and* $\Delta \mathrm{v} = \mathrm{tw}_{\mathrm{last}} - \mathrm{tw}_{\mathrm{first}}$*, the duration of the visit equals to the difference between the timestamp of the last and the first tweet of the visit.*

We built a multi-threaded crawler to access the Twitter Streaming API. The crawler collects the geo-tagged tweets filtered by location and processes the results to obtain a dataset representing a sequence of daily snapshots, with an average number of tweets per day greater than 40,000.

The dataset represents a sequence of daily snapshots, with an average number of tweets per day greater than 40,000. The data analysis reveals that the behavior of users is very heterogeneous: note the long tail of the probability distribution functions (PDF) both of the number of tweets and of the time interval that elapses between successive users' tweets. Figure 4.4(a) shows the PDF of the number of tweets per user in a month. Even if the volume of tweets per month is very high, most of the users (78%) post less than 10 tweets per month. This may depend on the fact that many users are tourists and then occasionally visit the city. 21% of users are more active making more than 10 tweets but less than 100. Finally very few users, just 1%, post more than 100 tweets per month. A similar pattern arises considering the time elapsed between successive tweets. Figure 4.4(b) shows that 60% of tweets are posted with an inter-tweet intervals less that one hour. In particular about 40% of these are posted with high frequency, i.e., with an inter time of 10 minutes. Moreover, the graph shows that only 28% of tweets are posted with a frequency greater than 3 hours. A similar trend is observed in [97], with about 46% of intervals between tweets greater than or equal to one hour.

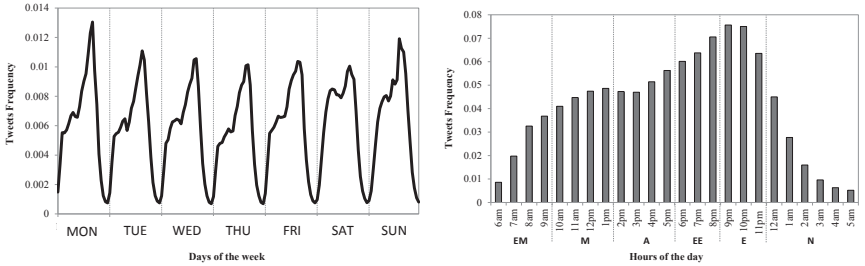


(a) (b)

Fig. 4.4. Probability Distribution Function of number of: monthly tweets per user (a); the time elapsed between consecutive tweets (b).

We also observe the tweets frequency during the course of the week. Figure 4.5(a) shows that the tweets rate for each day of the week has a periodic behavior. Days exhibit a peak in the evening and a dip at night time. The figure also highlights some differences between week days and the weekend. In particular, in weekends the volume of tweets is higher, mainly during the

morning and there is a peak at lunchtime. This is more evident on Saturdays. These patterns seem to mirror user behavior: for instance, during a week day, a user might spend morning and afternoon at the workplace, taking a lunch break in a restaurant, while in the evening he might go to the gym, to the cinema or stay at home. In order to exploit these temporal patterns for our classification task we divide the day into six different time slots, as shown in Figure 4.5(b).



(a) (b)

Fig. 4.5. Tweets frequency during a week (a) and its temporal evolution during the daily time slots (b).

The time slots, are formally specified by the following:

Definition 4.4. ***Time slots.*** TS *is a finite set of time slots with* $|\mathrm{TS}| = 6$. *Each* $\mathrm{ts} \in \mathrm{TS}$ *is a time-object of varying time duration belonging to a day.* $\mathrm{TS} = \{\mathrm{N}, \mathrm{EM}, \mathrm{M}, \mathrm{A}, \mathrm{EE}, \mathrm{E}\}$ *where:*

$\mathrm{N} = \mathrm{Night}[12:00\mathrm{am} - 05:59\mathrm{am}]$*;*
$\mathrm{EM} = \mathrm{EarlyMorning}[06:00\mathrm{am} - 09:59\mathrm{am}]$*;*
$\mathrm{M} = \mathrm{Morning}[10:00\mathrm{am} - 01:59\mathrm{pm}]$*;*
$\mathrm{A} = \mathrm{Afternoon}[02:00 - 05:59\mathrm{pm}]$*;*
$\mathrm{EE} = \mathrm{EarlyEvening}[06:00\mathrm{pm} - 08:59\mathrm{pm}]$*;*
$\mathrm{E} = \mathrm{Evening}[09:00\mathrm{pm} - 11:59\mathrm{pm}]$.

On the basis of this definition, we specify a mapping function TS(tw) to associate the corresponding time slot to the timestamp of a tweet.

4.2.2 Semantic Location Detection

The *semantic location detection* step consists of two main phases: (i) location detection and (ii) semantic category association to the detected locations so as to obtain *semantic locations*.

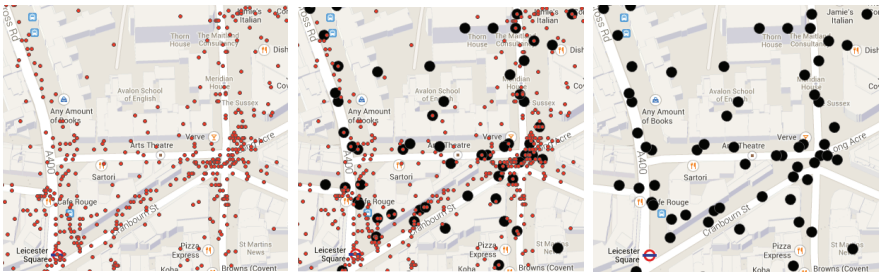
Location Detection

Due to the variable accuracy of GPS, a specific place in a city might be represented by slightly different GPS coordinates.

Thus, it is necessary to *cluster the locations* of the geo-tagged tweets so that each place is identified by a single pair of geographic coordinates. More precisely, we have used a dense clustering algorithm that receives as input a set of geographic coordinates L extracted from a set of geo-tagged tweets TW. Each location $l \in L$ may be represented by one or more pairs (lat, lon). What we want to achieve by applying the clustering algorithm is that each $l \in L$ is uniquely identified by a couple of geographical coordinates.
To this aim we extended OPTICS [98], an algorithm for finding density-based clusters in spatial data (although other algorithms could also have fulfilled the propose). We have exploited the implementation provided by ELKI [99], an open source data mining software focused on unsupervised methods in cluster analysis and outlier detection.

We adopt a tree-based model for the hierarchical dense clustering approach. The tree expresses the parent-children relationships of the nodes pertaining to different levels. Using a density-based clustering algorithm, we hierarchically cluster this geo-tagged data into some geo-spatial regions (set of clusters) in a divisive manner. The close geo-locations in the tweets from various users would be assigned to the same clusters on different levels. Thus, each level of the tree represents a different level of aggregation. More precisely, the leaves of the tree represents the geo-coordinates specified in the tweets. Then, starting from the leaves, the first level of the tree is obtained by applying dense clustering on the geo-coordinates of the tweets in order to obtain locations with the finer granularity detail. Given this hierarchical structure, a node on a level of the tree (that is a cluster of locations) represents a larger region that can be used to represent its descendant nodes obtaining this way a lower granularity in the location identification process. In other words, a location might belong to multiple clusters based on the different region scales it falls in.

The evolution of the clustering algorithm is shown in Figure 4.6.



(a) (b) (c)

Fig. 4.6. Evolution of Clustering Algorithm: the tweets geo-locations (a) are grouped by the algorithm so that each cluster respects spatial distance constraints (b). Each cluster of geo-locations is represented by its centroid (c).

The result of the clustering is a model composed of a set of clusters; each of such cluster corresponds to a geographic region that actually is a *dense region* visited by many users. A dense region can be defined as follows:

Definition 4.5. ***Dense Region.*** *A dense region stands for a geographic area of points that are densely visited by users, meaning that many tweets have been posted from that geographic area. It is, thus, characterized by a group of nearby GPS points and is represented by the GPS coordinated of its centroid point.*

The extend of the dense region depends on the granularity detail that we want to achieve with our analysis, ranging from a specific venue in the city (e.g., National Gallery, Big Ben, London Eye in London) to an area in the city (e.g., Trafalquasre in London) or a suburb in the city (e.g., Covent Garden in London) or a district in the city (zone 1 in London). The granularity of the geographical extension can be set through appropriate tuning of clustering parameters.
In this work we performed the *finest-grained analysis*. The geographical extend of the area is thus the one corresponding to a specific venue in the city like a restaurant, workplace, museum, etc. To detect significant locations in a city, the algorithm parameters are set such as to aggregate the geo-tagged locations at the extend of venues in the city. To this aim we used a clustering neighborhood radius of about 100 meters as it is reasonable to take this value for the spatial extension of typical locations in city environments. This setting of the cluster radius parameter enable us to find out some significant places such as museums, restaurants, shopping malls, etc, while ignoring geo-regions without semantic meaning, like the places where people wait for traffic lights or meet congestion.

Category Association

In most of the online social networks, a location is simply represented as latitude-longitude coordinates. However the semantics knowledge of the type of place (home, office, museum) a user is at is potentially very useful: it could allow to infer users common interests, to improve activity prediction and ultimately mobile user recommendation and advertisement.

Once the dense regions are detected we identify place semantics. We formulate the problem as a supervised learning task where the input vectors are designed according to: (i) a set of spatial-temporal features such as stay duration, time of day, place popularity, extracted from irregular social posts; and (ii) a Foursquare dataset retrieved by the Foursquare API. Specifically, we use the Foursquare categories as the ground truth class attribute of the category classifier. Later in this chapter (see Section 4.4), we will present the *Category Classifier* in detail.

We refer to the eight top categories of Foursquare, that are *Professional&OtherPlaces*, *College&University*, *Nightlife&Spot*,

Food, *Shop&Service*, *Arts&Entertainment*, *Outdoors&Recreation*, *Travel&Transport*. As compared to a raw GPS point, each location detected with our methodology carries a particular semantic meaning, such as the shopping malls we accessed or the restaurants we visited, etc. Moreover, we made a more fine grain classification, retrieving by the Foursquare API, the precise name of places, e.g. London Eye, Buckingham Palace, National Gallery and so on.

Example 4.6. Figure 4.7 shows an example of semantic location detection obtained through the proposed hierarchical dense clustering approach.

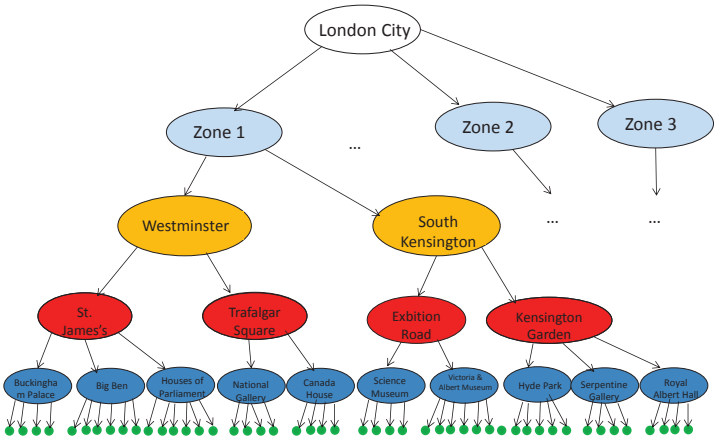


Fig. 4.7. Example of Hierarchical Clustering.

Analysing the Twitter dataset of London, with the semantic location identification phase we automatically labeled 70177 locations.

4.2.3 Travel Routes Generation

In this section we show how we can extract trajectories, also referred to as *travel routes*, from the collected geo-tagged tweets.

Roughly speaking a travel route is a temporally ordered sequence of places visited by users. In a day d a user u might visits one or more locations in a city. For each user we extract his daily travel routes. A daily travel route is defined as follows:

Definition 4.7. ***Travel Route.*** *A travel route is a sequence of locations visited by users during the same day according to temporal order. It is represented as a sequence of N pairs in the form (l_i, t_i) where $l_i = (x_i, y_i)$ and t_i is the timestamp. We can, thus, define a travel route as a spatial-temporal sequence of visited place:*

$$\mathrm{TR_{u,d}} = \mathrm{v_{l_0,t_0}} \longrightarrow \mathrm{v_{l_1,t_i}} \longrightarrow \ldots \longrightarrow \mathrm{v_{l_n,t_n}}$$

According to the tree model defined for the location identification process, we also obtain a hierarchy of travel routes. In particular, we connect the clusters of the same level according to the extracted trajectories: if consecutive places on one route are individually contained in two clusters, a link would be generated between the two clusters in a chronological direction according to the time serial of the two places.

Example 4.8. Referring to the example shown in Figure 4.7 we can derive the following travel routes with different granularity detail:
$TR_1 : BigBen \rightarrow NationalGallery \rightarrow ScienceMuseum \rightarrow HydePark$
$TR_2 : St.James's \rightarrow TrafalgarSquare \rightarrow ExbitionRoad \rightarrow KensingtonGarden$
$TR_3 : Westminster \rightarrow SouthKensington$

Considering the finer granularity detail we derive the travel route TR_1 among venues; the travel route TR_2 expresses the same travel path as TR_1 but the geo-coordinates are aggregated at the extend of suburbs whereas in travel route TR_3 the aggregation is at the extend of districts in London.

To the aim of this work we extract travel routes at the finest granularity detail, so travel routes among venues in the city. We extracted 455422 travel routes with an average cardinality of 3.31.

4.2.4 Mining popular Travel Routes

In this section we discuss one of the key point of our approach: the extraction of spatial-temporal patterns for the travel routes generated at the previous step. On the basis of this information we define then a set of features characterizing the travel routes.

Accordingly to most research in literature, trajectory patterns can be used as a concise description of frequent behaviors in terms of both space and time. Thus, the problem of trajectory pattern mining can be modeled as an extension of traditional sequential pattern mining and association rule mining. The idea is to determine sequences of places in the data, which occur together frequently in the data, and with similar transition times. The sequential pattern mining paradigm can be extended to this case by incorporating temporal constraints into successive elements of the sequence.
Given the travel routes generated at the previous step, the sequential pattern mining algorithm will find all the sequential patterns whose frequencies are no smaller than the minimum support. We refer to the mined frequent patterns as *trajectory patterns* or *frequent travel routes*. The frequency of a pattern is defined as the number of of frequencies subsuming the pattern.

We adopt a two-phase approach for mining popular travel routes: (i) the first phase consists of applying sequential pattern mining on the location sequences, also extracting (differently to many works in literature) locations sequences that are not continuous in corresponding original trajectories; (ii) the second one consists of extracting the maximal frequent sub-sequences from

all the frequent sequences obtained with the sequential pattern mining algorithm. This second step is necessary in order to ensure that trajectories with large segments in common are not reported simultaneously. To this aim we modify the well-known PrefixSpan [93] algorithm to obtain only maximal frequent patterns. In fact, a major challenge in mining frequent patterns from a large data set is the fact that such mining often generates a huge number of patterns satisfying the minimum support threshold, especially when the minimum support is set low. This is because if a pattern is frequent, each of its sub-patterns is frequent as well. A large pattern will contain an exponential number of smaller, frequent sub-patterns.

In the following is described a toy example of frequent travel routes mining.

Example 4.9. Let's suppose to extract from the tweets of London the following travel routes:

$T1 : MillenniumBridge \rightarrow StPaul'sCathedral \rightarrow BigBen \rightarrow TheLondonEye \rightarrow TowerBridge \rightarrow TateModern$

$T2 : BuckinghamPalace \rightarrow MadameTussauds \rightarrow BigBen \rightarrow TheLondonEye \rightarrow TowerBridge \rightarrow WestminsterBridge \rightarrow ParliamentSquare$

$T3 : BigBen \rightarrow TheLondonEye \rightarrow TrafalgarSquare \rightarrow TempleofMithras$

$T4 : BigBen \rightarrow TheLondonEye \rightarrow TowerBridge \rightarrow WestminsterBridge \rightarrow ParliamentSquare$

$T5 : BuckinghamPalace \rightarrow MadameTussauds \rightarrow M\&M'sWorld \rightarrow TowerBridge \rightarrow TateModern$

Given the above 5 travel routes the maximal frequent pattern mining algorithm determines the following frequent travel routes with a minimum support fixed at 2:

$BigBen \rightarrow TheLondonEye$ (frequency 5)

$TheLondonEye \rightarrow TowerBridge$ (frequency 4)

$TowerBridge \rightarrow TateModern$ (frequency 3)

$WestminsterBridge \rightarrow ParliamentSquare$ (frequency 2)

$BuckinghamPalace \rightarrow MadameTussauds$ (frequency 2)

Concerning our real evaluation of the London tweets, we mined 11959 patterns and then we extract only the maximal frequent patterns with a minimum support of 25, obtaining this way just 923 maximal frequent pattern to which we refer to as frequent travel routes. Of the 923 mined frequent travel routes, 451 concerns routes traveled by multiple users whereas 461 are routes traveled by only one user.

4.3 Statistical analysis of frequet Travel Routes

In this section we compute daily snapshots concerning the traveled routes, the visited places, the movements among them, and the users.

4.3.1 Spatial-temporal features of frequent travel routes

We define a set of features that exploit different information dimensions about users' movement. The features answer to several questions allowing to characterize the dynamics of human behavior and of their activity together with the featuring of cities and urban areas. Among such questions are also included the following ones: (1) how popular is a travel route? (2) is a travel route frequent? (3) how many popular location include the route? (4) how the travelers behave along the travel route?.

We denote the set of frequent travel routes in the Twitter dataset as $\mathcal{FTR}$, the set of category $\mathcal{C}$ and the set of users $\mathcal{U}$.

We first exploit global information about tweeets patterns of users going beyond a specific user. In this category we will include popularity and geographic features of travel routes and locations together with features that exploit transitions among venues. In this category we also define time aware features that capture information both on user activity in terms of visiting category of places but also temporal patterns of visits to specific places.

Number of journeys

The number of journeys along a travel route is the overall number of times that the route is traveled; it corresponds to the support of the frequent travel route ftr. It is actually the *frequency* of ftr and can be defined as follows:

$$\text{nJourney}(\text{ftr}) = |\{\text{ftr}_\text{i} \in \mathcal{FTR} : \text{ftr}_\text{i} = \text{ftr}\}| \tag{4.1}$$

Number of distinct travelers

The number of people who travel a route is indicative of its *popularity*. A user u is considered a traveler of a frequent travel route ftr if he travels at least once the trajectory. The number of travelers of a frequent travel route ftr can be expressed as follows:

$$\text{nTraveler}(\text{ftr}) = |\{\text{u} \in \mathcal{U} : \text{J}_{\text{ftr,u}} \neq \emptyset\}| \tag{4.2}$$

where $\text{J}_{\text{ftr,u}}$ is the set of journeys in the travel route ftr by user u. According to that we define a *popular travel route* as follows:

Definition 4.10. *A frequent travel route is popular when the number of distinct users traveling it is higher than a given threshold.*

This threshold is data dependent. In our case is fixed to 50.

Number of journeys per traveler

This feature represents the average number of times that distinct travelers take the route. This value is indicative of the periodicity of users behavior

along the trajectory, i.e. high values indicate that the travelers, on average, pass along the route many times. The feature is formalized as follows:

$$\text{JourneyPerUser(ftr)} = \frac{\text{nJourney(ftr)}}{\text{nTraveler(ftr)}} \tag{4.3}$$

Travel Route Category

This feature characterizes the travel route in terms of the category of the locations crossed in the route. Clearly, the category of the route could be just one or as many as the different categories to which the locations in the route belong. Formally can be defined as follows.

$$\mathcal{C}(\text{ftr}) = \{\text{c} : \forall \text{l} \in \text{ftr}, \text{Category(l)} = \text{c}\} \tag{4.4}$$

where $Category(l)$ represents a category of the location l.

Number of tweets

This feature is the number of total tweets that have been posted from locations in a travel route. This feature could give indication about the importance of the travel route.

$$\mathcal{TW}(\text{ftr}) = |\{(\text{u}, \text{l}, \text{t}) \in \mathcal{TW} : \forall \text{l} \in \text{ftr}\}| \tag{4.5}$$

Entropy of a travel route

This feature tells us how a travel route ftr is visited, whether users tend to travel regularly the route at usual times or they transit in it without any regularity. For this purpose we use the Shannon Entropy:

$$H(X_{ftr}) = -\sum_{u=1}^{n} p(x_u) log p(x_u), \quad where \;\; p(x_u) = f(ftr, u) \tag{4.6}$$

f(ftr, u) describes the distribution of the movements across the users, in other words it is the user's proportion of journeys at the route ftr and is defined as:

$$\text{f(ftr, u)} = \frac{|\text{J}_{\text{ftr,u}}|}{|\text{J}_{\text{ftr}}|} \tag{4.7}$$

where J_{ftr} is the whole set of journeys in the travel route.

We expect a small entropy for routes involving Professional or College&University places in which people tend to have more stable and periodic behavior. In contrast, a higher entropy value implies that many users do journeys along the route ftr, but they have very few journeys. This user behavior is typical along routes involving Arts&Entertainment or Nightlife&Spot places.

4.3.2 Experimental Evaluation

In this section we analyze the frequent travel routes obtained after the mining phase by exploring the features defined above. In particular, we focus on the 451 routes traveled by multiple users, obtained as detailed in Section 4.2.4.

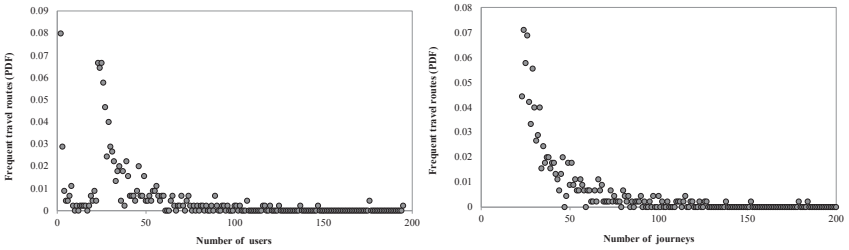
The overall analysis of traffic flows across time and space reveal that the majority of the frequent travel routes are tourist movement patterns, as the visited locations are London's most important historical and cultural sites. Moreover, the spatial-temporal information featuring the travel routes, like the number of users traveling the route (popularity), the number of journeys (frequency) along the route, and the regularity of users behavior (entropy), outline the profile of touristic trajectories.

The average length of the analyzed frequent travel routes is 2.4. This number is also confirmed in many studies in the literature as reported in the study [87] where the average length of touristic travel routes is 2, and in the study [94] where the authors explicitly say that it is not necessary to find out trajectories with long length as people would not visit many places in a trip and for this reason in that paper they work with 2-length trajectories.

As already explained, the number of users traveling a specific route is indicative of its *popularity*. Figure 4.8(a) shows the probability distribution function (PDF) of the number of distinct users per frequent travel route, highlighting that about 86% of travel routes are traveled by a number of distinct users greater than 10, with a peak when the number of user is 23. Among them, about 10% of trajectories is visited by more than 60 users.
More precisely, the graph shows a peak when the number of user is 23, in fact most of the trajectories (39%) are traversed by a number of users ranging from 23 to 30, 33% of them is visited by a number of users between 30 and 60, and 9% is visited by more than 60 users.
Figure 4.8(b) shows the PDF of the number of journeys per travel route. This value gives information about the *frequency* of travel routes. The graph highlights a trend similar to the previous distribution. In fact, more than 80% of the travel routes are traveled a number of times ranging from 10 to 60, while there is a few number of travel routes (about 16%) with a very large number of journeys; these last trajectories cross the most popular tourist attractions.

Even if the volume of journeys and users per travel route is quite high, we observe that the number of journeys per user is 1 or 2 for the 85% of travel routes. This depends on the fact that those travel routes are touristic and thus users occasionally travel across them, as shown for some trajectories in Table 4.1.

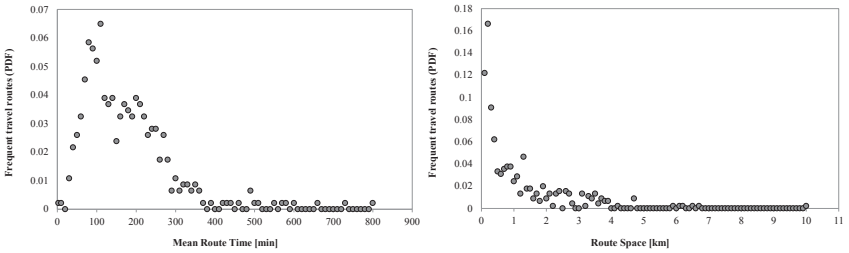
Figure 4.9(a) shows the mean time duration of frequent travel routes, referred to as *mean route time*. We consider mean values because the same route could be traveled several times and could start and finish in different daily time slots. Only 2% of travel routes are very fast route, in fact their duration is within 30 minutes. The graph shows that most of the travel routes



(a) (b)

Fig. 4.8. Probability Distribution Function of number of distinct users (a) and number of journeys (b) per frequent travel route.

(88%) last in a time period longer than half an hour and shorter than four hours and a half, with a pick around two hours. Finally, 12% of travel routes have a duration greater than four hours and a half.
One of the reasons why the trajectories have a temporal duration quite small is that 98% of travel routes are 2-length trajectories. Also, in 97% of travel routes, the sequences of visited locations is within the small radius of 4 km. In particular, 64% of trajectory has a *route space*, i.e. the distance in km between the first and the last location in the path, of about 1 km, as shown in Figure 4.9(b). Only 2% of travel routes covers a distance greater than 4 km, reaching a maximum of 13 km.



(a) (b)

Fig. 4.9. Probability Distribution Function of mean route time (a) and route distance (b) per frequent travel route.

Table 4.1 lists the 20 top frequent travel routes. We can notice that most of the frequent routes concerns visits to world famous historic sights, including Houses of Parliament, Big Ben, Buckingham Palace, and other London attractions like the London Eye and Harrods.
Table 4.1 contains 16 travel routes, referred to as *touristic travel routes*, characterized by a high number of distinct users traversing them (popular routes) and a high number of journeys along them. Important information is given by

Table 4.1. The TOP 20 frequent travel routes.

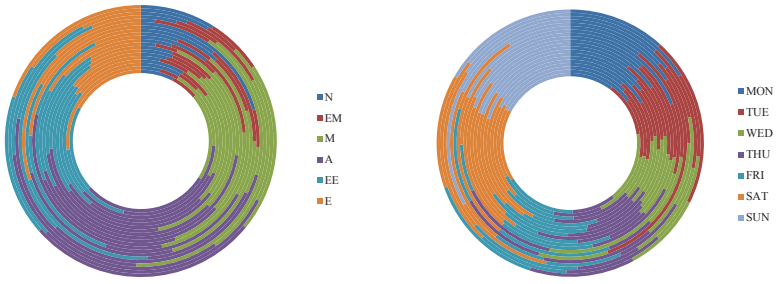
ID	Frequent travel route	Tweets	Users	Journeys	Journeys/ User	Entropy	Time [min]	Space [Km]
T0	Houses of Parliament → The London Eye	581	235	244	1	7.85	122.34	0.43
T1	The London Eye → Houses of Parliament	540	217	229	1	7.72	94.06	0.43
T2	Big Ben → The London Eye	490	195	205	1	7.58	97.96	0.46
T3	The London Eye → Big Ben	434	176	184	1	7.43	99.37	0.46
T4	Buckingham Palace → The London Eye	430	176	179	1	7.45	195.04	1.57
T5	The Tower of London → Tower Bridge	377	147	152	1	7.18	87.79	0.25
T6	Buckingham Palace → Houses of Parliament	318	137	138	1	7.09	187.28	1.26
T7	Houses of Parliament → Buckingham Palace	302	125	127	1	6.96	134.06	1.26
T8	Harrods → Hyde Park	299	120	126	1	6.88	211.24	1.09
T9	Tower Bridge → Tower of London	334	117	123	1	6.84	77.12	0.25
T10	Clapham Junction → ASDA Clapham Junction	533	2	119	60	0.07	114.56	0.24
T11	Trafalgar Square → National Gallery	276	116	118	1	6.85	60.04	0.08
T12	Red Lion → The Courtyard Theatre	374	2	115	58	0.07	173.33	0.22
T13	ASDA Clapham Junction → Clapham Junction	507	2	115	58	0.07	90.60	0.24
T14	06 St. Chad's Place → Burger King	483	3	114	38	0.14	93.84	0.29
T15	Hyde Park → Harrods	257	96	112	1	6.51	200.37	1.09
T16	Buckingham Palace → Trafalgar Square	274	107	111	1	6.72	142.34	1.22
T17	The London Eye→ Buckingham Palace	280	107	110	1	6.73	181.46	1.57
T18	The London Eye → Tower Bridge	254	103	105	1	6.68	195.84	3.08
T19	Tower Bridge → The London Eye	247	101	102	1	6.65	182.68	3.08

the high entropy values, indicating that the behavior of visitors is not regular, in fact they travel the routes on average only once, as confirmed by values in column *Journeys/Users*.

The table also contains the average time duration (mean route time) and the spatial distance (route space) of travel routes. We note that the travel routes last on average about 2 hours and a half and cover a mean radius of 1.1 km. Travel routes from The London Eye to Tower Bridge and vice versa, are the trajectories with the greatest distance between the visited locations. We observe that the district of Westminster presents the highest tourist density with travel routes covered by more than 170 users.

All these travel routes, could be used to plan an itinerary for a trip to London, providing information on attractions that are close/within walking distance of each other; grouping the most popular attractions by area to maximize time of tourists and to avoid them to crisscrossing all over the city; and advertising the public transportation system and walking routes. A different profile is presented by 4 trajectories in Table 4.1: T10, T12, T13 e T14, referred to as *residential travel routes*. These routes are traveled by 2-3 users who repeat often the paths, in fact the number of journeys is high, consequently they are frequent but not popular. In this case, the entropy is low because visitors run through these routes in a periodic way, returning on different days. For instance, T10 and T13 represent the movement from a railway station to a grocery store and vice versa. The users could be two employers of the store that arrive to work and come back to home by train.

Figure 4.10 shows the journeys temporal distribution of the 20 frequent travel routes, respectively during the hours of the day (a) and during the days of the week (b). Figure 4.10(a) shows that the 16 touristic travel routes are traveled for a short time period during night (N) and early morning (EM), while they are traveled mostly during morning (M) and afternoon (A), clearly this is justified by the fact that generally the touristic attractions of a city are visited during daylight hours. Moreover, many of those attractions (e.g., museums, historical buildings, art galleries) have daily admission and opening times. The daily time distribution is quite different for the 4 residential travel routes. For instance, T10 and T13 that include transport and work places, present a longer time interval during the early morning and the evening (E), maybe because in this time slots the users arrive or leave the work place, whereas T12 that consists of a movement between bar and pub, is traveled mainly during evening (E) and night (N).
Figure 4.10(b) presents a similar distribution of the number of journeys during the week for all the travel routes. The plot highlights that most of the routes have a greater number of journeys during the weekend and the lowest number of journeys on Wednesdays and Thursdays. There are few exceptions, for instance T12, that we consider a residential travel route, is traveled mostly during the week, in particular on Monday and Wednesday; likewise the residential travel route T14 is traveled mostly during Tuesday and a small number of times during Sunday. The graph also shows that T16 is the most traveled trajectory during Sunday, perhaps because a lot of tourists enjoy attend the notorious royal guard change and walking across the Mall up to Trafalgar Square.



(a) (b)

Fig. 4.10. Journey distribution of frequent travel routes during: the hours of a day (a); the days of a week (b).

Table 4.2 shows the 20 popular locations. As we expected, it contains the London's most famous tourist attractions. These locations are characterized by a high number of distinct visitors and by a high number of visits. The column *VisitPerDay* shows that the first 10 locations are visited daily a number

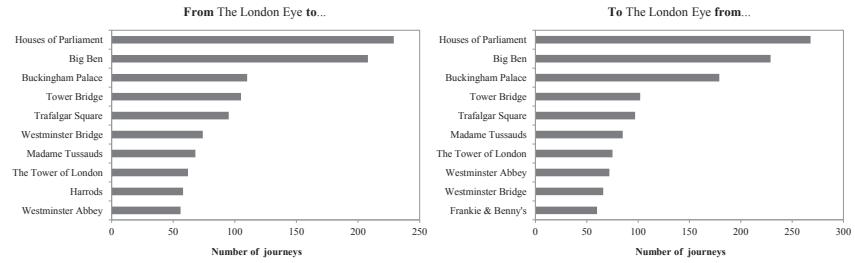
Table 4.2. The TOP 20 popular locations.

Popular location	Users	Visits	VisitPerDay	FrequencyFTR	Reachability	Incoming
The London Eye	1594	3323	21.58	52	24	25
Buckingham Palace	1284	2784	18.20	52	29	22
Harrods	996	1898	12.82	45	23	22
Houses of Parliament	922	1728	13.09	27	14	12
Trafalgar Square	890	1879	12.28	39	20	19
Tower Bridge	765	1419	9.72	27	12	15
Big Ben	737	1431	15.39	25	11	13
Hyde Park	650	1089	7.83	25	13	12
The Tower of London	539	988	7.06	22	12	10
Piccadilly Circus	501	894	11.92	21	9	12
Oxford Street	458	892	9.91	22	12	10
Westminster Abbey	438	815	5.82	16	9	7
St Paul's Cathedral	374	655	4.85	17	10	7
Selfridges & Co	355	607	4.71	13	9	4
M&M's World	337	493	3.79	15	5	10
Leicester Square	332	513	3.83	15	6	9
Frankie & Benny's	330	605	6.24	16	10	6
Tate Modern	302	433	3.52	11	6	5
London Pavilion	300	503	6.53	16	7	9
Westminster Bridge	278	434	3.29	9	5	4

of times greater than or equal to 10. It's interesting to note that the locations can belong to travel sequences of different frequent travel routes (FTR), even if few locations appear in a large number of different travel sequences. The *FrequencyFTR* column highlights that the locations can belong to travel sequences of different frequent travel routes (ftr), showing that the most popular locations presents high frequencyFTR. For instance, The London Eye and Buckingham Palace are in 52 distinct travel sequences, which is the maximum frequency value in the detected frequent travel routes. For each location, the table shows the number of other locations that can be reached from the one (*Reachability*), and the number of places from where the specific location can be reached (*Incoming*). In agreement with *FrequencyFTR*, The London Eye and Buckingham Palace are the places that allow to reach and are accessed by the highest number of places.

Focusing on The London Eye, we show in Figure 4.11(a) the top 10 locations that can be reached by that place and in Figure 4.11(b) the top 10 from which it is reached, considering the number of journeys between the places. The graphs highlight that the most frequent travel routes are those that go from The London Eye to Houses of Parliament, Big Ben, Buckingham Palace, Tower Bridge, Trafalgar Square, and vice versa, with differences in terms of the number of journeys of the routes in one direction than the other.

Figure 4.12 reports the reachability graph of the 10 top popular locations. The locations are represented as vertex connected by edges. A edge is a travel route between two locations, it has a direction and a weight that is the number of journeys along the specific direction. In the graph only the routes traveled more than 50 times are shown (the edges with a weight greater than 50).



(a) (b)

Fig. 4.11. The top 10 location: reachable from The London Eye (a); to reach The London Eye (b).

According to the values of Table 4.2, The London Eye and Buckingham Palace are the locations involved in the highest number of travel routes, thus they appear in the graph with the highest number of incoming and outgoing edges.

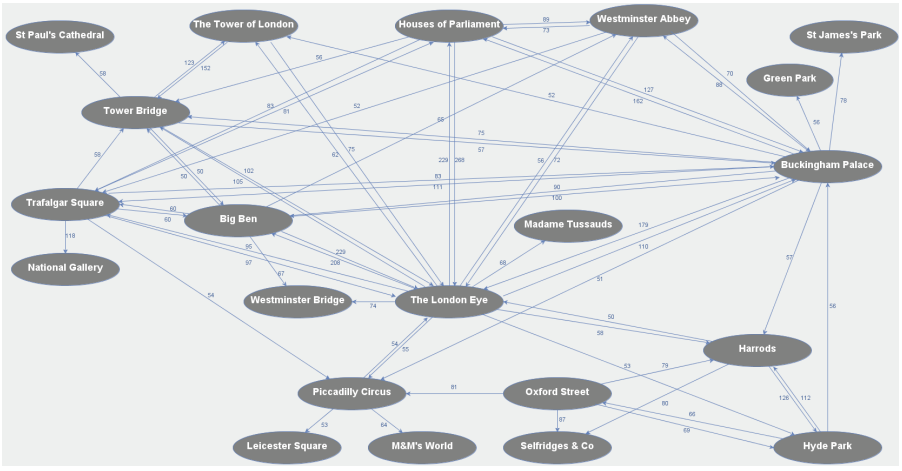
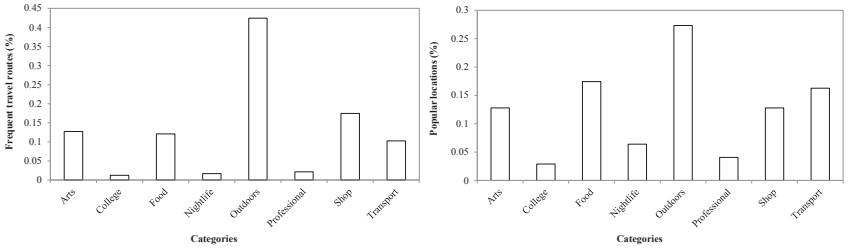


Fig. 4.12. Reachability Graph.

We determine place semantics according to Foursquare classification. Figures 4.13(a) and (b) show the categories distribution to which belong the frequent travel routes and the visited locations, respectively. Both the distributions highlight that the prominent activities of users are spending leisure time in the outdoor (e.g., park, square, mountain and river) or visiting landmarks (e.g. historic buildings, monuments, castles and gardens). In fact 42% of routes has travel sequences among *Outdoor&Recreation* locations (Figures 4.13(a)) . Those locations represent 27% of the overall visited places (Figures 4.13(b)). Over half of the travel routes (53%) visits places belonging to the categories

Food, *Shop&Service*, *Arts&Entertainment*, and *Travel&Transport*. Only 5% of routes visits work, educational and night-life locations. A similar pattern arises when considering the categories distribution of visited locations in Figures 4.13(b). This evidence reinforces our hypothesis that most of the travel routes are touristic movements.



(a) (b)

Fig. 4.13. Categories distribution of the frequent travel routes (a) and of visited locations (b).

4.4 Category Classifier

The problem of automatically associating semantics to geographical location is one which has attracted researchers attention as it enables a variety of rich applications ranging from recommendation, advertisement and better space planning. For instance, knowing if a place is of a certain category could help recommending targeted places, e.g., tourists could be recommended leisure places or monuments specifically. While, knowing the category of a place a user is in could be useful for automatically inferring activities of people, e.g., if a user is in a work place, he is likely working.
In this section we detail our location category inference task. Given a location in a city expressed exclusively as geographical coordinates, our aim is to label it on the basis of the spatial-temporal patterns exhibited by users in that location. To this aim, we have built a framework to infer the category of the visited places among a finite set of alternatives. We address the problem as a supervised classification task.
Formally, we can define the supervised classification problem of inferring the category of a place in the following way:

Given a location l *and its spatial-temporal patters, we aim to infer its category* c *among a finite set of categories* C.

4.4.1 Spatial - Temporal Patterns of Places

In this section we discuss one of the key point in our approach: the extraction of spatial-temporal patterns for each place. On the basis of this information we will define the set of machine learning features.
First of all we introduce the concept of consecutive tweets, based on the time slots defined above.

Definition 4.11. ***Consecutive tweets****. Two tweets* tw_1 *and* tw_2*, temporally ordered, are consecutive iff:*

$$\begin{aligned} TS(tw_1) = TS(tw_2) \vee TS(tw_2) = succ(TS(tw_1)) \\ \wedge \quad if \quad TS(tw_1) \neq N, \quad \delta(tw_1, tw_2) < 3h \end{aligned} \tag{4.8}$$

where $\mathrm{succ}(\mathrm{TS}(\mathrm{tw}_1))$ *is the successive time slots of* tw_1 *and* $\delta(\mathrm{tw}_1, \mathrm{tw}_2)$ *is the temporal distance between the tweets.*

This constraint guarantees that the time elapsed between tw_1 and tw_2 is not too long. If they are posted in the same time slot, the maximum temporal distance allowed is equal to the length of the time slot, that is limited. Moreover, if tw_1 and tw_2 are posted in successive time slots, the maximum distance allowed is three hours. If the time slot of the first tweet is night (N), we relaxed the constraint and the second tweet can be posted during the entire next slot, that is the early morning (EM). This choice is based on the observation that most people sleep during night, so we could not have tweets during that time period. Therefore, it is reasonable to think that in the early morning people are still in the place where they slept, even if there are no tweets to substantiate that.
In a day d a user u might visit one or more locations. For each user we compute his daily trajectories. A daily trajectory is formalized as follows:

Definition 4.12. ***Daily Trajectory****. A daily trajectory,* $\mathrm{DT}_{\mathrm{u,d}}$*, is a sequence of visits at different locations* $(\mathrm{v}_{\mathrm{l}_0}, \mathrm{v}_{\mathrm{l}_1}, \ldots, \mathrm{v}_{\mathrm{l}_n})$ *by a user* u *during a day* d*:*

$$\mathrm{DT}_{\mathrm{u,d}} = \mathrm{v}_{\mathrm{l}_0} \longrightarrow \mathrm{v}_{\mathrm{l}_1} \longrightarrow \ldots \longrightarrow \mathrm{v}_{\mathrm{l}_n}$$

When two consecutive tweets tw_1 and tw_2 of a user u are posted from two different locations l_1 and l_2, obviously the user has moved from one place to another. We assume that during the time interval between tw_1 and tw_2, referred to as movement time, $\mathrm{m}_{\mathrm{l}_1,\mathrm{l}_2}$, the user u was in l_1 (of course this is an assumption which does not factor in the transit time: we leave this improvement for future work).
If tw_1 and tw_2 are not consecutive (in other words, the time between them is too long) nothing can be assumed of the whereabouts of the user in that time frame, and we do not take into account the time interval between tw_1 and tw_2. The same applies when two not consecutive tweets are made in the same location l. In this case we consider the tweets as referring to two different visits to l.

A daily trajectory therefore contains distinct visits to locations; each visit to a location l, namely v_l, is characterized by the sequence of consecutive tweets. For each visited location l in $DT_{u,d}$ we compute the daily time spent in it by the user u during the day d, indicated with $tIn_l^{u,d}$, and formalized as follows:

$$tIn_l^{u,d} = \sum_{v_l \in DT_{u,d}} \Delta v_l + \sum_{v_{l'} \in DT_{u,d}} (m_{l',l}) \tag{4.9}$$

Where:

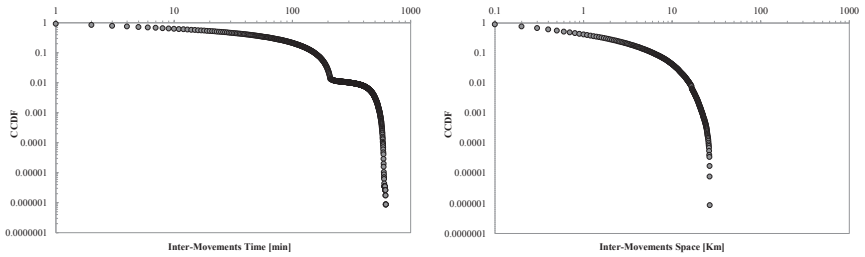
- $\sum_{v_l \in DT_{u,d}} \Delta v_l$ is the overall visits duration,
- $\sum_{v_{l'} \in DT_{u,d}} (m_{l',l})$ is the overall time for movements started from the location l by u in the day d.

Notice that, only the time intervals between two consecutive tweets, will be considered in the calculation of the visit duration.
To clarify the above, we show an example of a daily trajectory of user u:

$$DT_{u,d} = v_{l_1} \overset{(7,N)}{\longrightarrow} v_{l_2} \overset{(4,EM)(5,M)}{\longrightarrow} v_{l_2} \overset{(8,E)}{\longrightarrow} v_{l_3} \overset{(3,E)}{\longrightarrow}$$

For each visited location l is shown a list of couples; each couple is the number of consecutive tweets posted in l in the timeslot TS. In the example, the user visits 3 different locations. At night time (N) u posts 7 consecutive tweets in l_1, and moves during the early morning (EM) in l_2. The time that the user spends in l_1 is the period from the first tweet in N in l_1 and the first tweet in EM in l_2. u posts 4 tweets during EM and 5 tweets during M in l_2, and all the tweets are consecutive. After that, his next tweet is recorded too long after in the early evening (E). This means that the last tweet posted in M and the first posted in E are not consecutive. We cannot say that the user spends also the afternoon (A) and the early evening (EE) in l_2, so we consider two different visit to the location. From l_2 the user moves to l_3 during the evening (E). The time that the user spends at l_2 is composed of the sum of the time spent in it during the two visits, considering the consecutive tweets, plus the time elapsed between the last tweet in E in l_2 and the first tweet posted during E in l_3. Since we do not know where the user is after l_3, the time of the visit in l_3 is composed just of the difference between the last and the first tweet posted in it.
It is interesting to observe the trends of the temporal and spatial distance among the movements. Figure 4.14(a) shows the Complementary Cumulative Distribution Function (CCDF) of the temporal distance between the sequential tweets. The log-log plot presents a power-law functional form with a long tail indicating a wide variety of temporal measures. In fact, about 38% of movements occur frequently, 30% of them occur with an inter-time distance greater than 10 minutes but less than 1 hour, and another 30% of movements are less frequent, having a temporal distance that varies between 1 hour and 3 hours. Only a small percentage takes place after a period of time longer

than 3 hours. This can mean that users tend to tweet before they leave a place and when they reach a new location. Furthermore, this information is important for our work: this means that there are only a few cases that we cannot consider due to a temporal distance being too long.
Likewise, the distribution of the space distances that users travel when moving from one location to another is a power-low, as shown in Figure 4.14(b). In this case the CCDF highlights that most of users tend to move for short distances. In particular, 10% of movements take place within the small radius of 100 meters, 48% of movements cover a radius greater than 100 meters but within a distance of 1 km. Only 4% of movements affects a distance of more than 10 km.



(a) (b)

Fig. 4.14. Complementary Cumulative Distribution Function of: the inter-movements time (a); the inter-movements space (b).

4.4.2 Classification Class Attribute

To address the problem of automatic places labeling of an urban area we used a supervised learning approach. In this approach, the class attribute values for the dataset are known before running the algorithms. This data are called *labeled* data or *trining* data.
To this aim we use a Foursquare database of London, retrieved by the Foursquare API, as a source of a ground truth database of labels (any other database would have fulfilled the same purpose). The database of labeled places will be used by our classification algorithm to aid the geo-tweets classification. For the evaluation of this work we retrieve a mapping of each geo-tweet to the most-likely semantic location. We use these mapped locations for training our framework and to validate the prediction (more in the evaluation section). This database contains 39,304 Foursquare venues. For each venue we consider the pair (lat, lon) that identifies the location and the Foursquare category *4sq*. The main parameter of our mapping algorithm is the maximum mapping distance δ, that is the maximum allowable distance between a location to be mapped and a Foursquare location. Following different experiments we chose $\delta = 25$ meters. The output is a set of labeled clusters.

Many locations cannot be mapped with a Foursquare venue (mainly streets). Due to this, it is not possible to assign a category to all the clusters: in particular 58,119 clusters could not be associated to a category. As a result, we worked on 33,680 clusters. The distribution of places among the eight top categories of Foursquare, that are *Professional&OtherPlaces*, *College&University*, *Nightlife&Spot*, *Food*, *Shop&Service*, *Arts&Entertainment*, *Outdoors&Recreation*, *Travel&Transport*, is depicted in Figure 4.15. As expected, the category that includes the largest number of places is *Food*. *Shop&Service*, *Professional&OtherPlaces* and *Nightlife&Spot* have a substantial set of places, while the other categories include less than 10% of the total places.

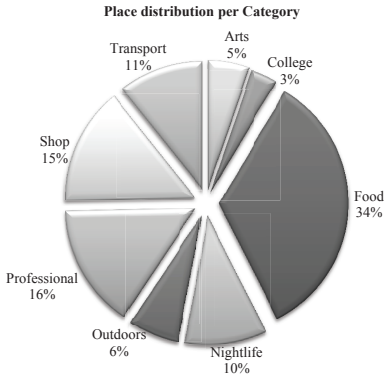


Fig. 4.15. Places distribution among the eight Foursquare categories.

4.4.3 Classification Features

We have i) labeled the locations through the database matching and ii) analyzed user behavior patterns at locations, so we aim to exploit those in our classifier to allow the labeling of unknown locations automatically. Our classifier must infer the category of a place based on the spatial-temporal patterns. In this section we derive a set of machine learning features that together with the ground truth places category will be the input for the supervised classification algorithms. The features answer the following questions: (1) *how many people* visit the place? (2) *how long*/ (3) *when*/ (4) *how is* the place visited? The features identified are listed below.

Number of visitors

Knowing the number of people who visit a place is indicative of its popularity. The most visited places in an urban area are the public places such as

squares and parks (*Outdoors&Recreation*), places of entertainment like museums and theaters (*Arts&Entertainment*), and certainly, all transport hubs such as subways, railway stations (*Travel&Transport*). The number of visitors of a location l, is the cardinality of U_l, defined above, that includes the distinct users who have posted at least one tweet while at the location.

$$\text{nVisitors}_l = |U_l| \tag{4.10}$$

Daily User Stay

One of the most important aspects to capture is the daily time that users spend in a place. People spend on average 60% to 65% of their time at home and between 20% and 25% at work or college [104]. In accordance to this, we expect that the time that users spend in a work place (*Professional&OtherPlaces*) and in an educational place (*College&University*) is longer than the time elapsed in other places like bar or restaurant (*Food*), boutique or Internet cafe (*Shop&Service*), train or tube station (*Travel&Transport*). For each user u and location l we measure the average daily time spent in l by u, ($\text{dTime}_{l,u}$). We compute the average daily time among all the users that visit the location, and is defined as:

$$\begin{aligned} \forall u \quad \text{dTime}_{l,u} &= \sum_d \text{tIn}_l^{u,d} / \text{nD}_{l,u} \\ \text{DailyUserStay}_l &= \frac{\sum_u \text{dTime}_{l,u}}{|U_l|} \end{aligned} \tag{4.11}$$

Where $\text{tIn}_l^{u,d}$ is the time that a user u spent in l on day d, as expressed in equation 4.9, and $\text{nD}_{l,u}$ is the number of days in which u visits l.

Short and Long Time Visit

We define a pair of machine learning attributes that provide information on the duration of the single visits. Our aim is to separate short from long visits. As an example, a user could visit the same bar several times within a day (maybe the bar near his work place). All the visits are very short except one during the lunch break. The daily time will be the sum of all the time periods spent in that place, which can be quite long in relation to the time spent for having lunch, even if most of the visits to this place are short. Furthermore, a place could be visited by people with different profiles. For instance, a post office could be visited by a lot of short stay customers, and by a small number of employees that spend all the working day there. We define the short visits feature as:

$$\begin{aligned} \text{shortVisits}_l &= |v \in V_l : \Delta v < 30\text{min}| \\ \text{ShortTimeVisit}_l &= \frac{\text{shortVisits}_l}{|V_l|} \end{aligned} \tag{4.12}$$

Where Δv is the time length of the visit, and $|V_l|$ is the total number of visits in l as defined above. Similarly, the long visits feature is described as:

$$\begin{aligned} \text{longVisits}_l &= |v \in V_l : \Delta v > 3h| \\ \text{LongTimeVisit}_l &= \frac{\text{longVisits}_l}{|V_l|} \end{aligned} \tag{4.13}$$

Night Location

This feature is proposed to separate the locations visited mostly at night time from those visited during the day. This is formally achieved by measuring the ratio between the night time (nigthlyTime_l) and the total time spent at a location. It is expected that this value will be greater for nightclubs and pubs (*Nightlife&Spot*), or places such as bowling or theaters (*Arts&Entertainment*), rather than places belonging to categories like *Professional&OtherPlaces*, *College&University*, and *Shop&Service* categories.

$$\begin{aligned} \text{nigthlyTime}_l &= [t_0, t_1] : t_0 \in EE \wedge t_1 \in N \\ \text{NightlyLocation}_l &= \frac{\text{nightlyTime}_l}{\text{tIn}_l} \end{aligned} \tag{4.14}$$

Weekend Location

Likewise, we aim to partition the locations visited mostly during the weekend from those visited during weekdays. We expect that the time spent in a location during the weekend (weekendTime_l) will be higher for categories of places such as *Nightlife&Spot*, *Outdoors&Recreation*, *Arts&Entertainment*. Conversely, weekdays are mainly working days: the most popular places will be those of category *Professional&OtherPlaces* and *College&University*.

$$\begin{aligned} \text{weekendTime}_l &= [t_0, t_1] : t_0 \in \text{Sat} \wedge t_1 \in \text{Sun} \\ \text{WeekendLocation}_l &= \frac{\text{weekendTime}_l}{\text{tIn}_l} \end{aligned} \tag{4.15}$$

These metrics allow us to keep into account: the number of visitors at each location, how long the place is visited and when the visits occur. We now formalize how a place is visited. In this sense we must consider whether the location is a transient or a steady place, i.e., if users visit a location at usual times or they transit in it without any regularity. For this purpose we use the Shannon Entropy (see Equation 4.6) to measure the uncertainty of three variables:

- $H(T_l)$, tweets in the location l;
- $H(F_l)$, frequency of the tweets in the location l;
- $H(D_l)$, days in the location l.

We now describe these in detail.

Tweets Entropy

This feature tells whether users tend to tweet regularly in a location l, describing the distribution of its tweets across the users. For the feature $H(T_l)$, $f(l, u)$ is the user's proportion of tweets at the location l and is defined as:

$$f(l,u) = \frac{|TW_{l,u}|}{|TW_l|} \tag{4.16}$$

Where $TW_{l,u}$ is the set of tweets posted in l by user u, while TW_l is the whole set of tweets posted from that location. We expect a small entropy in the *Professional&OtherPlaces* or *College&University* places in which people tend to have more stable and periodic behavior. In contrast, a higher entropy value implies that many users tweet from l, but they have very few tweets. This user behavior is typical in *Arts&Entertainment* or *Nightlife&Spot* places.

Frequency Tweet Entropy

Another important aspect to be highlighted is the tendency of users to tweet frequently (or less frequently) from a place. It is expected that in the workplace users do not often post messages (*Professional&OtherPlaces* or *College&University*). On the other hand, during leisure time, while sightseeing or visiting a museum, the frequency of tweets could be greater: someone could post photos or information about the visited place (*Arts&Entertainment*, *Outdoors&Recreation*, *Shop&Service* or *Travel&Transport*). The feature $H(F_l)$ is expressed formally by defining $f(l, u)$ as the user's proportion of frequent tweets from location l:

$$f(l,u) = \frac{|\tau_{l,u} : \tau_{l,u} < 15min|}{|\tau_l|} \tag{4.17}$$

Where $\tau_{l,u}$ is the set of short time intervals (i.e., shorter than 15 minutes) elapsed between two tweets posted from l by user u, while τ_l is the set of all the inter-tweets times at l.

Daily Visit Entropy

This feature is correlated to tweets entropy, but it further emphasizes users regularity for location l. The feature captures whether users visit a location in a periodic way, returning on different days. $H(D_l)$ is defined as:

$$f(l,u) = \frac{nD_{l,u}}{nD_l} \tag{4.18}$$

Where $nD_{l,u}$ is the number of days in which a user u visits the location l, while nD_l is the total number of days in which l is visited, as defined above. For instance, in a work or educational place (*Professional&OtherPlaces* or *College&University*), users might exhibit periodic behavior (e.g., five days a

week), whereas leisure or food places may attract many opportunistic visits. In the following we show how a classifier can be trained over the features extracted to associate place categories to geographical coordinates automatically.

4.4.4 Classification Tasks

The features defined in the previous section are used to construct classification tasks, using a supervised approach. Foursquare categories are fed to the classifier, together with the feature vector. We have used six classification algorithms: J48, Decision Table, Multilayered Perceptron, Bayesian Network, K* and LogitBoost. The algorithms are those included in the collection of Weka, a popular open source data mining toolkit [58].
In order to estimate the accuracy of our prediction algorithms, we used the 10-fold cross-validation as model validation technique and the set of metrics typically used in classification problems: precision (P), recall (R), and f-measure (FM). The overall task consists of assigning a decision class label (the category) to a set of unclassified locations, described by the defined set of features. *Our purpose is to infer the category of locations in a city knowing a set of categories* C *that may be associated with those locations.*

Supervised learning algorithms

We have used six classification algorithms, each one based on a different data mining classification technique.
We focused on two groups of *logical learning algorithms*, *decision trees* and *rule-based classifiers*, focusing on the J48 in reference to the first method, and on the Decision Table as regards the second. The *J48* generates a pruned or unpruned C4.5 decision tree as a predictive model, in which the internal nodes denote the different attributes, the branches between the nodes represent the possible values that these attributes can have in the observed samples, while the terminal nodes tell us the classification value [3]. The *Decision Table* model consists in a decision table with a rule mapping to the majority class. The set of features that are included in the table are named schema, while the labelled instances from the space defined by the schema is the body of the table. Given an unlabeled instance, a decision table classifier perform a hierarchical breakdown of the data, with two attributes at each level of the hierarchy, trying to extract matches. The classification value if different from the majority class of the table if the set of matching instances is not empty. In that case, the classification values is the majority class of this set [5].
Other well-known algorithms are based on the *perceptron-based techniques*. In this field, we chose the *Multilayered Perceptrons* which try to solve the weaknesses of the perceptrons that can only classify linearly separable sets of instances. It is a multi-layer neural network, composed of an interconnected group of nodes called neurons, and the synapses store parameters called

weights. The first step is to determine input-output mapping, by training the network on a set of paired data. Then, the weights of the connections between neurons are fixed and the network is used to determine the classifications of a new set of data [100].
Among the *statistical learning algorithms*, we select the *Bayesian Network* classifier, a directed acyclic graph that allow an effective representation of the joint probability distribution over a set of random variables. Each vertex in the graph represents a random variable, and edges represent direct correlations between the variables. The main assumption is that every leaf in the network (every attribute) is independent from the rest of the attributes, given the state of the root in the network (the class variable) [101].
Another category is *instance-based learning* that consist of lazy-learning algorithms, as they delay the induction or generalization process until classification is performed. We run the K^* that classify an instance by comparing it to a database of pre-classified examples. The fundamental assumption is that similar instances will have similar classifications. It uses an entropy-based distance function to define similar instance [102].
The last algorithm that we consider is the *LogitBoost*. It is a *meta-learning technique* and performs additive logistic regression, in fact carries out classification using a regression scheme as the base learner [103].

Binary problem

In this formulation of the problem the cardinality of C is one. This means that we are interested in finding if a place as isolated by the GPS coordinates of a tweet (and then clustered as described in Section 4.2.2) belongs to a certain category or not. Formally, this problem results in a binary classification problem:

Given $\mathrm{c} \in \mathrm{C}$*, is* c *the category of a location or not?*

For each category we construct an annotated training dataset, in which we have two groups of instances. One group belongs to the category c and all its elements are labeled with *Yes*, while the other group is a random sample of the population. In the latter the instances are labeled with *No*. Figure 4.16 shows the average accuracy achieved for each Foursquare category by the six classifiers. The highest average accuracy is obtained for categories *Professional&OtherPlaces* and *College&University*. We can label the work places with an accuracy of 90%. *College&University* has average accuracy at 81%, followed by *Outdoors&Recreation* averaging 67%. *Arts&Entertainment* and *Travel&Transport* have moderate average accuracy at 63%. For the classes *Food*, *Nightlife&Spot* and *Shop&Service*, we observe that the average accuracy is slightly less than 60%. The latter value may be justified from the diversity of places that each of these classes represents. For instance, *Food* includes bars and restaurants. Not only the time spent in a bar is generally shorter than the time spent in a restaurant, but also, people tend to visit a bar at

different times during a day, while a restaurant is visited mainly at lunch time and dinner time. For the purpose of recommendation, this level of category clustering however seems quite sufficient. On the contrary, work places tend to be visited according to a unique temporal pattern.

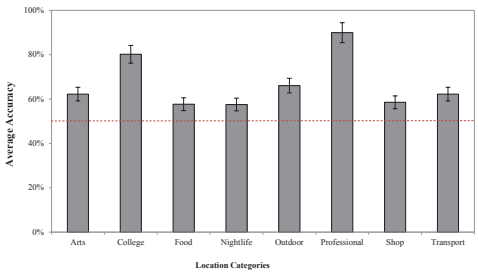


Fig. 4.16. Average Accuracy of the eight location categories inferred by the six classifiers.

Table 4.3 shows the values of the validation metrics precision (P), recall (R), and f-measure (FM) of the most performing classification algorithms for each category class. As evidenced by the values in the table, the accuracy is very high for the class *Professional&OtherPlaces*, with f-measure of 92%. This measure is the harmonic mean of precision and recall, each of which assumes high values for that class. The recall is 1.0: this means that every work place was labeled correctly as belonging to this class. This is also confirmed by the precision value where 86% of the items labeled as *Professional&OtherPlaces* do indeed belong to that class, and a small percentage of other items were incorrectly labeled as work place. Similarly, 89% of *College&University* was labeled correctly, and only about 22% of other places are classified as educational locations. On the whole, the recall value for the other categories is on average around 70%, except for *Food* and *Nightlife&Spot* that presents misclassification for nearly half the time.

Table 4.3. Precision (P), Recall(R) and F-Measure(FM) of the most performing classification algorithms for each of the eight location categories.

Category	P	R	FM
Art&Entertainment	0.64	0.68	0.66
College&University	0.78	0.89	0.83
Food	0.58	0.48	0.52
Nightlife&Spot	0.59	0.53	0.56
Outdoors&Recreation	0.67	0.69	0.68
Professional&OtherPlaces	**0.86**	**1.00**	**0.92**
Shop&Service	0.58	0.68	0.63
Travel&Transport	0.63	0.70	0.66

Figure 4.16, for each category shows the error bars that correspond to standard deviations in the performance of the six classifiers over each class. These bars are short, meaning that the experimental measurements dispersion of all the classifiers on a specific category is small. This is particularly evident looking at the plot in Figure 4.17, where all the classifiers tend to infer well the same categories as *Professional&OtherPlaces* and *College&University*. The accuracy of the classifiers varies according to the category to be predicted. In particular, the most performing classifier is the LogitBoost that achieves the best accuracy for four categories: *College&University*, *Food*, *Professional&OtherPlaces* and *Outdoors&Recreation*. It is followed by J48 that is the best classifiers for *Arts&Entertainment* and *Travel&Transport*, and by DecisionTable that classifies better *Nightlife&Spot* and *Shop&Service*. All of these learners have an average accuracy of 68%, followed by the Bayesian Network classifier and the Multilayered Perceptron, with an average accuracy of 67%. The less performing if the K* with 64% of average accuracy.

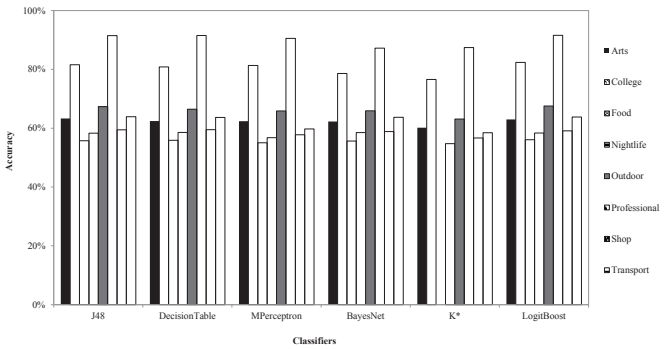


Fig. 4.17. Accuracy of the six classifiers over the eight location categories in solving the Binary Problem.

Classifiers are able to better discriminate the categories of places where users behavior is more stable and regular. Moreover, the accuracy rate is high for categories in which it is possible to identify a predominant user pattern.

Profiling problem

A more difficult problem is to choose the category of a place among a set of categories. The problem is formulated as a multinomial classification problem:

Given the set of category C*, which is the category* $c \in C$ *of a location?*

We consider three categories of places representative of the different human activities that typically characterize the everyday life of people. These categories are *Food&Drink*, *Leisure* and *Work*. In particular, the category

Food&Drink includes all the places in which people eat and drink, like day-night bars and cafes (e.g., coffee shop, Whisky Bar, Wine Bar), restaurant (e.g., Italian, Chinese, Pizza), pub and so on. This category is the union of the Foursquare categories *Food* and *Nightlife&Spot*. A *Leisure* place is a place in which people spend time free from the demands of work or duty, where one can rest, do shopping, travel, enjoy hobbies or sports. This category groups the Foursquare categories *Arts&Entertainment*, *Outdoors&Recreation*, *Shop&Service* and *Travel&Transport*. Conversely, the category Work includes place in which people spent time in business, work, and education. It includes the Foursquare categories *College&University* and *Professional&OtherPlaces*. The results of our analysis show that the aggregated sets present similarities between their spatial-temporal patterns. For instance, the visits at recreation places, art and entertainment locations present similar properties, attracting opportunistic visitors, generally in similar time period. On the contrary, the time that users spend in the work and educational places is different and longer. Moreover, public areas and transportation points, as well as food places are visited by a greater number of people, without periodic behavior, than the work places. As an evidence of that mentioned above, the following Figures 4.18 and 4.19 show two features: the daily time that users spend in a place (Daily User Stay), and the average number of visitors per place, of all the Foursquare categories.

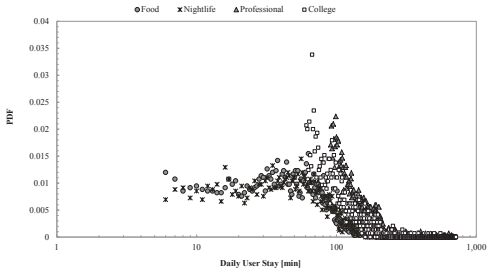


Fig. 4.18. Probability Distribution Function of the daily time that users spent in Food, Nightlife&Spot, College&University and Professional&OtherPlaces locations.

Figure 4.18 reports on the PDF of Daily User Stay, of the pair of categories *Food* and *Nightlife&Spot*, grouped in *Food&Drink*, and the couple *College&University* and *Professional&OtherPlaces*, aggregated in *Work*. Notice that *Food* and *Nightlife&Spot*, appear with similar temporal distributions, in which half of places have a mean daily time of stay of less than 60 minutes, and another 30% between 60 and 90 minutes. Conversely, *College&University* and *Professional&OtherPlaces* exhibit longer times to stay: in all this kind of locations the users spend at least 1 hour.
Figure 4.19 clearly shows that the places associated with the category *Food&Drink* (i.e., *Food* and *Nightlife&Spot*) are visited by an av-

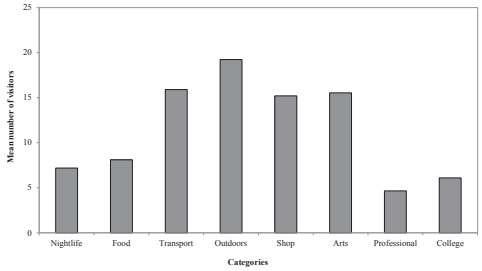


Fig. 4.19. Mean number of visitors for the eight location categories.

erage of 7-8 users, while the Leisure places (i.e., *Arts&Entertainment*, *Outdoors&Recreation*, *Shop&Service* and *Travel&Transport*) have a number of users exceeds 15, and finally in the Work places (i.e., *College&University* and *Professional&OtherPlaces*) the number of visitors has averaged 5.
The input of the classifiers is an annotated training dataset, in which we have three groups of instances, each one belonging to one category set and labeled either with *Food&Drink*, *Leisure* or *Work*. In solving the profiling problem, we used the six mining algorithms, as for the binary problem. In Figure 4.20 it can be observed that the labels can be predicted with a mean accuracy of about 64%. The highest performing classifier is LogitBoost with an accuracy of 66%, while BayesNet and the K* discriminate instances with accuracy slightly exceeding the 60%.

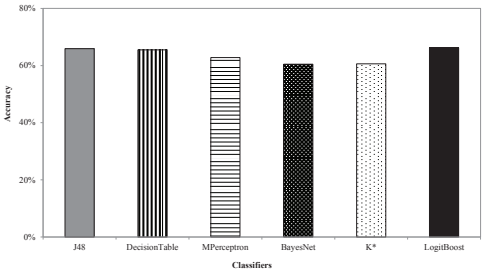


Fig. 4.20. Accuracy of the six classifiers in solving the Profiling problem.

Table 4.4 shows the values of precision (P), recall (R), and f-measure (FM) as they result from the LogitBoost classifier. The precision values indicate that for the categories *Food&Drink* and *Work* about 70% of items are correctly labeled as belonging to the correct class. *Leisure* presents a precision of 64%, and it has a good recall equals to 72%, overcome by *Work* that has the best recall value. In particular, 84% of work and educational places is identified as *Work*. On the contrary, the proportion of places belonging to *Food&Drink* that are correctly identified as such is not very high, this because they are

confused with *Leisure* places. This is a consequence of the fact that the two classes present some similarities: for instance, both the categories of places attract visitors in similar time period, e.g., Casino belonging to *Leisure* and Nightclub belonging to *Food&Drink* are visited during night time.

Table 4.4. Precision (P), Recall(R) and F-Measure(FM) of Food&Drink, Leisure and Work.

Category	P	R	FM
Food&Drink	0.71	0.36	0.48
Leisure	0.64	0.72	0.68
Work	0.68	0.84	0.75

4.5 Conclusions and future work

In this chapter we presented a novel methodology to extract and analyze the time- and geo-references associated with social data so as to mine information about human dynamics and behaviors within urban context. We performed a fine grain analysis of unpredictable and irregular information coming from geo-tagged tweets. The main contributions are listed below.

- A methodology to extract spatial-temporal patterns from geo-tweets.
- Place semantics detection based purely on spatio-temporal features such as stay duration, time of day and of week of the typical stay, place popularity, and regularity of users behavior in the place. We designed two classification tasks for place labeling:
  - The **Binary problem** aims to infer whether a place belongs to a certain category. Results demonstrate that we can correctly answer with an average accuracy of about 70%, reaching the highest accuracy for work and educational places (90% and 80%, respectively). This classification can be useful for instance to decide if a specific ad which is related to food promotions is appropriate for a location (i.e., only if the location is a restaurant or a bar).
  - The **Profiling problem** allows to infer the category of a place among a set of categories. In this study, we consider three different category sets that typically characterize people's everyday life. Our results show that with an accuracy of 66% we can identify if a place is one in which people eat and drink, or it is a leisure place, or a place for work/study.
- A statistical approach to build a real-life dataset of frequent travel sequences and interesting locations. In particular, we extracted a set of daily trajectories and used a sequential pattern mining algorithm to discover frequent travel routes. We then, defined a set of spatial-temporal features over such routes and, accordingly, performed a statistical characterization of patterns, rules and regularities in moving trajectories.

Future work include the integration of such methodology in a recommender system for trip planning, personalized navigation services, and location-based services useful for urban planning and management.

5

Conclusion

The very large use of mobile devices with more advanced computing capability, and ubiquitous access to the web from anytime and anywhere, have significantly raised the interest in Mobile Computing.
Mobile devices are becoming an essential part of human life, allowing people to accumulate rich experience in ubiquitous web access by using Android phones and tablets, iPhones, iPads, and other portable devices. Those devices are pervasively connected to global network as well as to persons.
One important aspect to note about the recent use of mobile computing is the increased access to social networking services, that are being regularly used by individuals, groups, friends, colleagues etc. where they share information, communicate with each other by means of using websites like Facebook, Twitter, and many more. As a consequence, social networking have migrated to mobile platform.
As a result of the increasing internet connectivity and the rapid adoption of mobile devices, a huge number of mobile web-based services and location-aware services are appearing in many real-life mobile scenarios, such as healthcare, vehicle control, wireless security systems, risk management, and navigation software to guide users in unfamiliar places and on tours.
However, many of the applications envisioned for mobile computing place novel demands on software systems. Much research is needed to overcome technological obstacles in low power system design, sustainable battery life, unreliability and limited bandwidth of the wireless channel, relatively small storage space, mobility, and security in a wireless environment.
The main aim of the carried out research work is designing techniques and methodologies that allow professionals to develop mobile services by optimally utilizing the limited resources and to take advantage of the extensive amount of available data to aid the automatic characterization of the dynamics of the urban environment.

As a first result of this study we presented the Energy-Aware (EA) scheduling strategy that assigns computational tasks over a network of mobile devices optimizing the energy usage.

We considered a multi-hop wireless ad-hoc mobile network architecture in which mobile nodes cooperate in a peer-to-peer style and are organized into local groups (also termed clusters or mobile groups). The main design principle of our scheduler is to find a task allocation in order to prolong the network lifetime and maximizes the number of alive devices by balancing the energy load among devices.
To this end, the EA scheduler implements a two-phase heuristic-based, decentralized approach. This algorithm first tries to assign a task locally to the cluster that generated the execution request by maximizing the cluster residual life. If the task cannot be assigned locally, the second phase of the algorithm is performed by assigning the task to the most suitable node all over the network of clusters, maximizing this way the overall network lifetime.
We performed a complete and exhaustive experimental evaluation of our scheduling strategy, considering heterogeneous scenarios. The results assessed that the EA strategy is able to extend the network lifetime and the number of alive devices compared to alternative scheduling strategies, while meeting application level performance constraints. In other words, EA addressed both the energy constraints of battery-operated wireless devices and the decentralized nature of mobile networks demanded by distributed computing over mobile environments.

As a second result of our work, we described a methodology that aims to analyze social geo-tagged data to discover people and community behavior. The basic assumption is that people often follow similar routes: e.g., they go to work every day traveling the same roads.
The proposed methodology first of all require the collection of geo-tagged post through Twitter. Due to the slightly variable accuracy of GPS, the locations of the geo-tagged tweets have been clustered to identify each place by a single pair of geographic coordinates. For a better understanding of users patterns we associated a semantic label to places visited by users. We then mined frequent travel routes using a sequential pattern mining algorithm. In the next steps we extracted spatial-temporal information for each of them and performed a statistical characterization of patterns, rules and regularities in moving trajectories.
Thanks to our approach we provided the top interesting locations and frequent travel sequences among these locations, in a given geo-spatial region. Thus, if we have enough data to model typical behaviors, such knowledge can be used to predict and manage future movements of people.

Learning lessons and feature work

The analysis and evaluation presented in this thesis highlight the importance of mobile devices in many human activities and pointed out how research activities are needed to make them efficient and useful. In particular, we investigated how the new generation of location-based mobile services can play a significant role in the urban computing for serving people and cities.

As we have seen in our experiments, the first major challenge is that we deal with the sparse and irregular Twitter data. This is more complex to deal with, compared to the analysis of data coming from GPS traces of mobile devices or taxis, or from Call Detail Records recorded by the BTS towers, that are the data sources of most of the existing works.
The second challenge is that, starting from this unpredictable data, our methodology is able to extract the venues in a city. We do not just consider the collection of the tweets geo-locations but we cluster the GPS coordinates to detect places in a city. It's important to note that our methodology achieves very fine grained clustering. The goal is not to identify regions but precise venues in a urban area like bars, restaurants, parks, theaters, museums, school and so on. However our cluster algorithm has some weaknesses, for instance, if we have a building (a single GPS coordinates or cluster) with multiple places, a restaurant in the ground floor and offices in the rest, the system is not able to deal with this situation. We have left this as a future work.
We mentioned earlier that knowledge on the semantics of geo-tagged venues can provide effective data representations to understand space. Our classifier performs the hard task of automatically infer places category, not without encountering limitations. Even if it achieves a good average accuracy, we notice that the classifier is able to better discriminate the categories of places where users behavior is more stable and regular. Moreover, the accuracy rate is high for categories in which it is possible to identify a predominant user pattern. In other words, some types of places are easier to predict than others. This may depend on a number of factors, such as the fact that the information retrieved by tweets has inherent limitations with respect to the inference task, or, from a data mining perspective, that the features we have introduced in the present work are biased towards a specific set of classes. As future work we aim to introduce features that better discriminate classes with spatial-temporal similarity, and we wish to include the study of different Twitter users profiles based on the patterns discovered with our technique.
A challenge that may be worth exploring in future work is the analysis of the users sharing contents to define new features.
Another aspect we want to point out is that the proposed methodology is general and it can be applied to other datasets. Obtaining better or worse results mainly depends on frequency of tweets. If we apply our methodology using a dataset in which users tweet more frequently we expect to have a better performance. On the contrary, if we consider a dataset with a low volume of significant daily snapshots, we expect that the quality of the analysis will decrease. We plan to extend this work presenting different case studies on datasets from different cities with different characteristics and in different periods.
Finally, future work includes the integration of such methodology in a recommender system for trip planning, personalized navigation services, and location-based services useful for urban planning and management.

Acknowledgements

During the time of writing of this PhD thesis I received support and help from many people. In particular, I am thankful to my supervisor, Prof. Domenico Talia, who was very generous with his time and knowledge and assisted me in each step to complete my PhD course.
I would like to thank also the people of my research group, especially Carmela Comito, who helped closely during my study period. Thanks are also due to Paolo Trunfio and Marco Lackovic for their intellectual support, ideas and feedback.
I am grateful to Prof. Jon Crowcroft and Cecilia Mascolo, who have given me the privilege to work with them in Cambridge.
To my colleagues and friends Sandra, I met in Cambridge, Chiara and Mariateresa goes my deepest gratitude for their encouragement and friendship.
And finally, but not least, thanks go to my whole family and friends, especially my Mum, my Dad and my sister Anna Carmen, who have been an important and indispensable source of moral support, and Pierluigi, whose precious presence has been vital to the achievement of this goal, giving me motivation whenever I needed it.

La presente tesi è cofinanziata con il sostegno della Commissione Europea, Fondo Sociale Europeo e della Regione Calabria. L'autore è il solo responsabile di questa tesi e la Commissione Europea e la Regione Calabria declinano ogni responsabilit sull'uso che potr essere fatto delle informazioni in essa contenute.

References

1. Perkins C E (2001) Ad Hoc Networking. Addison-Wesley.
2. Giordano S (2002) Mobile Ad Hoc Networks. In Handbook of Wireless Networks and Mobile Computing (ed I. Stojmenović), John Wiley and Sons Inc., New York, USA.
3. Quinlan J R (1993) C4.5: Programs for Machine Learning.
4. Rappaport T S (2002) Wireless Communications: Principles and Practices. 2nd Edition, Prentice Hall.
5. Kohavi R (1995) The power of decision tables. ECML, 174-189.
6. Muller N J (1998) Mobile Telecommunications factbook. McGraw-Hill, New York.
7. Satyanarayanan M (1996) Fundamental challenges in mobile computing. In Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing (PODC), 1–7, New York, USA.
8. Satyanarayanan M (2010) Mobile computing: the next decade. In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (MCS).
9. Mascolo C (2010) The Power of Mobile Computing in a Social Era. Internet Computing, IEEE 14(6):76–79.
10. Silberschatz A, Galvin P B, Gagne C (2006) Sistemi operativi. Concetti ed esempi. 7th Edition.
11. Hiltz S R, Turoff M (1993) The Network Nation 2. Addison-Wesley.
12. Zheng Y, Zhou X (2011) Location-Based Social Networks: Users. In Computing with Spatial Trajectories. Springer.
13. Gao H, Liu H (2014) Data Analysis on Location-Based Social Networks. In Mobile Social Networking, 165–194, Springer New York.
14. Flinn J, Satyanarayanan M (1999) Energy-aware adaptation for mobile applications. Symp. on Operating Systems Principles, 48–63.
15. Flinn J, Narayanan D, Satyanarayanan M (2001) Self-Tuned Remote Execution for Pervasive Computing. In Procceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII).
16. Kravets R, Krishnan P (1998) Power Management Techniques for Mobile Communication. International Conference on Mobile Computing and Networking (MobiCom'98).

17. Cano J C, Manzoni P (2000) A Performance Comparison of Energy Consumption for Mobile Ad Hoc Network Routing Protocols. Eighth IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'00), pp.57.
18. Perkins C, Royer E (2000) Ad Hoc On-Demand Distance Vector (AODV) Routing. Internet Draft, MANET Working Group, draft-ietfmanet-aodv-05.txt.
19. Johnson D B, Maltz D A, Broch J (1999) The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet Draft, MANET Working Group, draft-ietf-manet-dsr-03.txt.
20. Park V, Corson m, (1999) Temporally-Ordered Routing Algorithm (TORA) Version 1 - Functional SpecificatioN. Internet Draft, MANET Working Group, draft-ietf-manet-tora-spec-02.txt.
21. Perkins C E, Bhagwat P (1994) Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. Computer Communication, 234–244.
22. Chun B G, Iannaccone G, Iannaccone G, Katz R, Gunho L, Niccolini L (2009) An Energy Case for Hybrid Datacenters. HotPower.
23. Petrucci V, Loques O, Moss D (2009) A dynamic configuration model for power-efficient virtualized server clusters. In 11th Brazillian Workshop on Real-Time and Embedded Systems (WTR).
24. Petrucci V, Loques O, Niteroi B, Moss D (2009) Dynamic configuration support for power-aware virtualized server clusters. In WiP Session of the 21th Euromicro Conference on Real-Time Systems. Dublin, Ireland.
25. Liu L, Wang H, Liu X, Jin X, He W, Wang Q, Chen Y (2009) GreenCloud: a new architecture for green data center. In 6th international conference industry session on Autonomic computing and communications, 29–38. ACM New York, USA.
26. Tesauro G, Das R, Chan H, Kephart J, Levine D, Rawson F, Lefurgy C (2007) Managing power consumption and performance of computing systems using reinforcement learning. Advances in Neural Information Processing Systems.
27. Das R, Tesauro G, Kephart K O, Levine D W, Lefurgy C, Chan H (2008) Autonomic multi-agent management of power and performance in data centers.
28. Berral J L, Goiri I, Nou R, Juli F, Guitart J, Gavald R, Torres J (2010) Towards energy-aware scheduling in data centers using machine learning. Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, e-Energy '10, Passau, Germany, 215–224.
29. Hamscher V, Schwiegelshohn U, Streit A, Yahyapour R (2000) Evaluation of job-scheduling strategies for grid computing. Lecture Notes in Computer Science, 191–202.
30. Lu Y, Benini L, Micheli G (2000) Low-Power Task Scheduling for Multiple Devices. In International Workshop on Hardware/Software Codesign.
31. Weiser M, Welch B, Demers A, Shenker S (1994) Scheduling for Reduced CPU Energy. In Proceedings of First Symposium on Operating Systems Design and Implementation.
32. Hu J, Marculescu R (2004) Energy-Aware Communication and Task Scheduling for Network-on-Chip Architectures under Real-Time Constraints. In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, Paris, France.

33. Hu J, Marculescu R (2003) Energy-aware mapping for tile-based noc architectures under performance constraints. Proceedings of the ASP-DAC 2003. Asia and South Pacific, 233-239.
34. Hu J, Marculescu R (2005) Energy- and performance-aware mapping for regular noc architectures. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, 24(4):551-562.
35. Topcuoglu H, Hariri S, Wu M J (2002) Performance-effective and lowcomplexity task scheduling forheterogeneous computing. IEEE Transactions on Parallel and Distributed Systems, 13(3):260-274.
36. Luo J, Jha N K (2003) Power-profile driven variable voltage scaling for heterogeneous distributed real-time embedded systems. In VLSI Design, 369-375.
37. Luo J, Peh L S, Jha N (2003) Simultaneous dynamic voltage scaling of processors and communication links in real-time distributed embedded systems. In Design, Automation and Test in Europe Conference and Exhibition, 1150-1151.
38. Yan L, Luo J, Jha N K (2005) Joint dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 24(7):1030-1041.
39. Luo J, Jha N K (2000) Power-conscious joint scheduling of periodic task graphs and aperiodic tasks in distributed real-time embedded systems. ICCAD.
40. Somasundara A A, Ramamoorthy A (2007) Mobile Element Scheduling with Dynamic Deadlines, in IEEE transactions on Mobile Computing, 6(4).
41. Gu Y, Bozda D, Ekici E, zgner F, Lee C G (2005) Partitioning Based Mobile Element Scheduling in Wireless Sensor Networks. Proc. of IEEE SECON.
42. Gurun S, Krintz C (2003) Addressing the energy crisis in mobile computing with developing power aware software. UCSB, Computer Science Department, Technical Report.
43. Rudenko A, Reiher P, Popek G, Kuenning G (1999) The Remote Processing Framework for Portable Computer Power Saving. In Proceedings of the ACM Symposium on Applied Computing.
44. Li K, Kumpf R, Horton P, Anderson T (1994) A Quantitative Analysis of Disk Driver Power Management in Portable Computers. USENIX conference, 279–292.
45. Zhuo J, Chakrabarti C (2005) An efficient dynamic task scheduling algorithm for battery powered DVS systems. ASP-DAC'05, 846-849.
46. Zhang Y, Hu X, Chen D (2002) Task scheduling and voltage selection for energy minimization. DAC'02, 183-188.
47. Seth K, Anantaraman A, Mueller F, Rotenberg E (2003) FAST: Frequency-Aware Static Timing Analysis. IEEE RTSS, 40–51.
48. Aydin H, Melhem R, Moss D, Mejia-Alvarez P (2004) Power-Aware Scheduling for Periodic Real-Time Tasks. IEEE Transaction on Computers, 53(5):584600.
49. Alsalih W, Akl S G, Hassanein H S (2005) Energy-Aware Task Scheduling: Towards Enabling Mobile Computing over MANETs. IPDPS05, 242a.
50. Feeney L M. Investigating the energy consumption of an IEEE 802.11 network interface. Technical report T99:11, SICS.
51. Garey R, Johnson D (1975) Complexity bounds for multiprocessor scheduling with resource constraints. SIAM J. Computing, 4:187-200.
52. Chang H W D, Oldham W J B (1995) Dynamic task allocation models for large distributed computing systems. TPDS 6:1301-1315.

53. Comito C, Falcone D, Talia D, Trunfio P (2011) Energy Efficient Task Allocation over Mobile Networks. Proc. of the 9th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2011), Sydney, Australia, 380-387.
54. Comito C, Falcone D, Talia D, Trunfio P (2013) Efficient Allocation of Data Mining Tasks in Mobile Environments. Concurrent Engineering: Research and Applications, 21(3):197–207.
55. Comito C, Falcone D, Talia D, Trunfio P (2012) A distributed allocation strategy for data mining tasks in mobile environments. Proc. of the 6th International Symposium on Intelligent Distributed Computing (IDC 2012), Calabria, Italy, Studies in Computational Intelligence, 446:231–240, Springer.
56. Comito C, Falcone D, Talia D, Trunfio P (2013) Scheduling Data Mining Applications in Mobile Computing Environments. ERCIM News, 93:15–16.
57. Comito C, Talia D, Trunfio P (2011) An Energy-Aware Clustering Scheme for Mobile Applications. IEEE Scalcom'11, 15-22.
58. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten I H (2009) The weka data mining software: An update. SIGKDD Explor. Newsl., 11(1):10–18.
59. developer.android.com
60. Kwak H, Lee C, Park H, Moon S (2010) What is Twitter, a social network or a news media?. Proceedings of the 19th international conference on WWW, 591–600, ACM, New York, USA.
61. Lindqvist J, Cranshaw J, Wiese J, Hong J, Zimmerman J (2011) Im the mayor of my house: examining why people use foursquare-a social-driven location sharing application. In: Proceedings of the 2011 annual conference on Human factors in computing systems, 2409-2418, ACM.
62. Cho E, Myers S, Leskovec J (2011) Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1082-1090, ACM.
63. Yamaguchi Y, Amagasa T, Kitagawa H (2013) Landmark-Based User Location Inference in Social Media. In COSN'13 Proceedings of the first ACM conference on Online social networks, 223–234.
64. Scellato S, Mascolo C, Musolesi M, Latora V (2010) Distance matters: Geo-social metrics for online social networks. In: Proceedings of the 3rd conference on Online social networks. USENIX Association.
65. Brown C, Nicosia V, Scellato S, Noulas A, Mascolo C (2012) Where online friends meet: Social communities in location-based networks. In: Sixth International AAAI Conference on Weblogs and Social Media.
66. Eisenstein J, O'Connor B, Smith N A, Xing E P (2010) A Latent Variable Model for Geographic Lexical Variation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Cambridge, MA.
67. Ahmed A, Hong L, Smo A (2013) Hierarchical Geographical Modeling of User Locations from Social Media Posts. In WWW13.
68. Fujisaka T, Lee R, Sumiya k (2010) Exploring urban characteristics using movement history of mass mobile microbloggers. In Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications, 13-18, ACM.
69. Levashkin S, Bertolotto M (2011) GeoSpatial Semantics. Lecture Notes in Computer Science, 6631:108123. Springer, Berlin / Heidelberg.

70. Ben-Akiva M, Bierlaire M, Koutsopoulos H, Mishalani R (1998) Dynamit: a simulation-based system for traffic prediction. In: DACCORS Short Term Forecasting Workshop, The Netherlands.
71. Cheng Z, Caverlee J, Lee K (2010) You are where you tweet: a content-based approach to geolocating twitter users. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, 759-768, ACM.
72. Gao H, Tang J, Liu H (2012) Modeling geo-social correlations for new checkins on location-based social networks. 21st ACM International Conference on Information and Knowledge Management.
73. Chang J, Sun E (2011) Location 3: How users share and respond to location-based data on social networking sites. Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media.
74. Cheng Z, Caverlee J, Lee K, Sui D (2011) Exploring millions of footprints in location sharing services. In: Proceedings of the Fifth International Conference on Weblogs and Social Media.
75. Noulas A, Scellato S, Mascolo C, Pontil M (2011) An empirical study of geographic user activity patterns in foursquare. Proceeding of the 5th International AAAI Conference on Weblogs and Social Media.
76. Li N, Chen G (2009) Analysis of a location-based social network. In International Conference on Computational Science and Engineering, 4:263-270, IEEE.
77. Kurashima T, Iwata T, Irie G, Fujimura K (2010) Travel route recommendation using geotags in photo sharing sites. In Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM '10), 579–588. ACM, New York, USA.
78. Lu X, Wang C, Yang J M, Pang Y, Zhang L (2010) Photo2trip: generating travel routes from geo-tagged photos for trip planning. In Proceedings of the international conference on Multimedia (MM '10), 143–152. ACM, New York, USA.
79. Cao H, Mamoulis N, Cheung D W (2005) Mining frequent spatiotemporal sequential patterns. In Proceedings of ICDM.
80. Mamoulis N, Cao H, Kollios G, Hadjieleftheriou M, Tao Y, Cheung D W (2004) Mining, indexing, and querying historical spatiotemporal data. In Proceedings of KDD.
81. Girardin F, Dal Fiore F, Ratti C, Blat J (2008) Leveraging Explicitly Disclosed Location Information to Understand Tourist Dynamics: A Case Study. Journal Location Based Serv. 2(1):41–56.
82. Cranshaw J, Toch E, Hong J, Kittur A, Sadeh N (2010) Bridging the gap between physical location and online social networks. In Ubiquitous Computing, 119-128.
83. Volkovich Y, Scellato S, Laniado D, Mascolo C, Kaltenbrunner A (2012) The length of bridge ties: Structural and geographic properties of online social interactions. In ICWSM.
84. Wakamiya S, Lee R, Sumiya K (2011) Urban area characterization based on semantics of crowd activities in twitter. In GeoSpatial Semantics, 108-123.
85. Crandall D J, Backstrom L, Huttenlocher D, Kleinberg J (2009) Mapping the worlds photos. In WWW, 761-770.
86. Noulas A, Scellato S, Lathia N, Mascolo C (2012) Mining user mobility features for next place prediction in location-based services. In ICDM, 1038-1043.

87. Yin Z, Cao L, Han J, Luo J, Huang T S (2011) Diversified trajectory pattern ranking in geo-tagged social media. In SDM, 980-991.
88. Gabrielli L, Rinzivillo S, Ronzano F, Villatoro D (2014)From tweets to semantic trajectories: Mining anomalous urban mobility patterns. In Citizen in Sensor Networks, 26-35.
89. Sakaki T, Okazaki M, Matsuo Y (2010) Earthquake shakes twitter users: Real-time event detection by social sensors. In WWW, 851-860.
90. Takahashi T, Abe S, Igata N (2011) Can twitter be an alternative of real-world sensors? In Human-Computer Interaction. Towards Mobile and Intelligent Interaction Environments, pages 240-249.
91. Giannotti F, Nanni Y, Pinelli F, Pedreschi D (2007) Trajectory pattern mining. In KDD, 330-339.
92. Cesario E, Comito C, Talia D (2013) Towards a cloud-based framework for urban computing, the trajectory analysis case. In CGC, 16-23.
93. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu M C (2001) Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In ICDE.
94. Zheng Y, Zhang L, Xie X, Ma W Y (2009) Mining interesting locations and travel sequences from gps trajectories. In WWW, 791-800.
95. Chen Z, Chen Y, Wang S, Zhao Z (2012) A supervised learning based semantic location extraction method using mobile phone data. In CSAE, 548–551.
96. Krumm J, Rouhana D (2013) Placer: Semantic place labels from diary data. UbiComp, 163–172.
97. Chalmers D, Fleming S, Wakeman I, Watson D (2011) Rhythms in twitter. SocialObjects.
98. Ankerst M, Breunig M M, Kriegel H P, Sander J (1999) Optics: Ordering points to identify the clustering structure. SIGMOD Rec., 28(2):49–60.
99. Achtert E, Bernecker T, Kriegel H P, Schubert E, Zimek A (2009) Elki in time: Elki 0.2 for the performance evaluation of distance measures for time series. SSTD, 436–440, 2009.
100. Rumelhart D E, Hinton G E, Williams R J (1986) Parallel distributed processing: Explorations in the microstructure of cognition. chapter Learning Internal Representations by Error Propagation, 318-362.
101. Friedman N, Geiger D, Goldszmidt M, Provan G, Langley P, Smyth P (1997) Bayesian network classifiers. In Machine Learning, 131-163.
102. Cleary J G, Trigg L E (1995) K*: An instance-based learner using an entropic distance measure. In Machine Learning, 108-114.
103. Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting. Annals of Statistics, 28:2000.
104. Montoliu R, Blom J, Gatica-Perez D (2013) Discovering places of interest in every- day life from smartphone data. Multimedia Tools and Applications, 62(1):179–207.
105. Lyytinen K, Yoo Y (2002) Issues and challenges in ubiquitous computing. Communications of the ACM 45(12):62-65.
106. Lyytinen K, Yoo Y, Varshney U, Ackerman M S, Davis G, Avital M, Robey D, Sawyer S, Sorensen C (2004) Surfing the next wave: Design and implementation challenges of ubiquitous computing environments. Communications of the AIS (13):697-716.
107. Deepak G, Pradeep B S (2012) Challenging Issues and Limitations of Mobile Computing. IJCTA, 3(1):177–181.

108. Mascolo C, Capra L, Emmerich W (2002) Mobile computing middleware. Lecture Notes In Computer Science, 2497:20–58. Springer-Verlag New York, USA.
109. Sorathia K, Joshi A (2009) My World Social Networking through Mobile Computing and Context Aware Application. IMC, CCIS 53:179-188, Springer-Verlag Berlin Heidelberg.
110. Tamilarasi M, Chandramathi S, Palanivelu T G (2008) Efficient energy management for mobile ad hoc networks. UbiCC Journal, 3(5).
111. Lefurgy C, Rajamani K, Rawson F, Felter W, Kistler M, Keller T (2003) Energy Management for Commercial Servers Computer, 36(12):39–48.
112. Bianchini R, Rajaniony R (2004) Power and energy management for server systems Computer. Long Beach, CA, 37(11):68–76.
113. Pinheiro E, Bianchini R, Carrera E, Heath T (2001) Load balancing and unbalancing for power and performance in cluster-based systems. In Workshop on Compilers and Operating Systems for Low Power, 180:182–195.
114. Chase J, Anderson D, Thakar P, Vahdat A, Doyle R (2001) Managing energy and server resources in hosting centers. 18th ACM symposium on Operating systems principles (SIGOPS), 35(5):103–116.
115. Filani D, He J, Gao S, Rajappa M, Kumar A, Shah R, Nagappan R (2008) Dynamic Data Center Power Management: Trends, Issues and Solutions. Intel Technology Journal.
116. Dondo R, Cerd J (2007) A Cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. European Journal of Operations Research 176:1478–1507.
117. L. M. Feeney (2001) An energy-consumption model for performance analysis of routing protocols for mobile ad hoc networks. Mobile Networks and Applications Journal, 6(3):239–250.
118. Liao L, Fox D, Kautz H (2007) Extracting Places and Activities from GPS Traces Using Hierarchical Conditional Random Fields. The International Journal of Robotics Research, 26(1):119–134.
119. Barnes S, Scornavacca E (2004) Mobile marketing: the role of permission and acceptance. International Journal of Mobile Communications 2(2), 128-139.
120. Barwise P, Strong C (2002) Permission-based mobile advertising. Journal of interactive Marketing 16(1), 14-24.
121. Dia H (2001) An object-oriented neural network approach to short-term traffic forecasting. European Journal of Operational Research 131(2), 253-261.
122. Goswami D (2013) Mobile Computing. International Journal of Advanced Research in Computer Science and Software Engineering, 3(9).
123. Thormann B (2005) Modeling of dynamic resource allocation in a network on chip. MA Thesis, Royal Institute of Technology (KTH), Stockholm, Sweden.
124. Smith B (1982) Reflection and Semantics in a Procedural Programming Langage. Phd thesis, MIT.