UNIVERSIDAD POLITÉCNICA DE CARTAGENA

DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

ADVANCE IN OPTIMAL DESIGN AND DEPLOYMENT OF AMBIENT INTELLIGENCE SYSTEMS

PABLO A. LÓPEZ-MATENCIO PÉREZ

2012

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

ADVANCE IN OPTIMAL DESIGN AND DEPLOYMENT OF AMBIENT INTELLIGENCE SYSTEMS

PABLO A. LÓPEZ-MATENCIO PÉREZ

ADVISOR
JAVIER VALES ALONSO

2012

To Concha and Marcos

Abstract

A brilliant future is forecasted for Ambient Intelligence (AmI) systems. These comprise sensitive environments able to anticipate people's actions, and to react intelligently supporting them. AmI relies on decision-making processes, which are usually hidden to the users, giving rise to the so-called smart environments. Some of those envisioned environments include smart homes, health monitoring, education, workspaces, sports, assisted living, and so forth.

Moreover, the complexity of these environments is continuously growing, thereby increasing the difficulty of making suitable decisions in support of human activity. Therefore, decision-making is one of the critical parts of these systems. Several techniques can be efficiently combined with AmI environments and may help to alleviate decision-making issues. These include classification techniques, as well as mathematical programming tools.

In the first part of this work we introduce two AmI environments where decision-making plays a primary role:

- An AmI system for athletes' training. This system is in charge of monitoring ambient variables, as well as athletes' biometry and making decisions during a training session to meet the training goals. Several techniques have been used to test different decision engines: interpolation by means of (m, s)-splines, k-Nearest-Neighbors and dynamic programming based on Markov Decision Processes.

- An AmI system for furtive hunting detection. In this case, the aim is to locate gunshots using a network of acoustic sensors. The location is performed by means of a hyperbolic multilateration method.

Moreover, the quality of the decisions is directly related to the quality of the information available. Therefore, is necessary that nodes in charge of sensing and networking tasks of the AmI infrastructure must be placed correctly. In fact, the placement problem is twofold: nodes must be near important places, where valuable events occur, and network connectivity is also mandatory. In addition, some other constraints, such as network deployment cost could be considered. Therefore, there are usually tradeoffs between sensing capacity and communication capabilities. Two kinds of placement options are possible. Deterministic placements, where the position for each node can be precisely selected, and

random deployments where, due to the large number of nodes, or the inaccessibility of the terrain, the only suitable option for deployment is a random scattering of the nodes.

This thesis addresses three problems of network placement. The first two problems are not tied to a particular case, but are applicable to a general AmI scenario. The goal is to select the best positions for the nodes, while connectivity constraints are met. The options examined are a deterministic placement, which is solved by means of an Ant Colony Optimization metaheuristic for continuous domains, and a random placement, where partially controlled deployments of clustered networks take place. For each cluster, both the target point and dispersion can be selected, leading to a stochastic problem, which is solved by decomposing it in several steps, one per cluster.

Finally, the third network placement scenario is tightly related to the furtive hunting detection AmI environment. Using a derivate-free descent methodology, the goal is to select the placement with maximal sensing coverage and minimal cost. Since both goals are contrary, the Pareto front is constructed to enable the designer to select the desired operational point.

Resumen

Se ha pronosticado un futuro excepcional para los sistemas de Inteligencia Ambiental (AmI). Dichos sistemas comprenden aquellos entornos capaces de anticiparse a las necesidades de la gente, y reaccionar inteligentemente en su ayuda. La inteligencia de estos sistemas proviene de los procesos de toma de decisión, cuyo funcionamiento resulta transparente al usuario. Algunos de estos entornos previstos pertenecen al ámbito de los hogares inteligentes, monitorización de la salud, educación, lugares de trabajo, deportes, soporte en actividades cotidianas, etc.

La creciente complejidad de estos entornos hace cada vez más difícil la labor de tomar las decisiones correctas que sirvan de ayuda a los usuarios. Por tanto, la toma de decisiones resulta una parte esencial de estos sistemas. Diversas técnicas pueden utilizarse de forma eficaz en los sistemas AmI para resolver los problemas derivados de la toma de decisiones. Entre ellas están las técnicas de clasificación, y las herramientas matemáticas de programación.

En la primera parte de este trabajo presentamos dos entornos AmI donde la toma de decisiones juega un papel fundamental:

- Un sistema AmI para el entrenamiento de atletas. Este sistema monitoriza variables ambientales y biométricas de los atletas, tomando decisiones durante la sesión de entrenamiento, que al atleta le ayudan a conseguir un determinado objetivo. Varias técnicas han sido utilizadas para probar diferentes generadores de decisión: interpolación mediante (m, s)-splines, k-Nearest-Neighbors, y programación dinámica mediante Procesos de Decisión de Markov.

- Un sistema AmI para detección de caza furtiva. En este caso, el objetivo consiste en localizar el origen de un disparo utilizando, para ello, una red de sensores acústicos. La localización se realiza utilizando el método de multilateración hiperbólica.

Además, la calidad de las decisiones generadas está directamente relacionada con la calidad de la información disponible. Por lo tanto, es necesario que los nodos de la infraestructura AmI encargados de la obtención de datos relevantes del usuario y del ambiente, estén en red y situados correctamente.

De hecho, el problema de posicionamiento tiene dos partes: los nodos deben ubicarse cerca de los lugares donde ocurren sucesos de interés, y deben estar conectados para que

los datos capturados sean transmitidos y tengan utilidad. Adicionalmente, pueden considerarse otras restricciones, tales como el coste de despliegue de red. Por tanto, en el posicionamiento de los nodos es habitual que existan compromisos entre las capacidades de sensorización y de comunicación. Son posibles dos tipos de posicionamiento. Posicionamiento determinista donde puede seleccionarse de forma precisa la posición de cada nodo, y, aleatorio donde debido a la gran cantidad de nodos o a lo inaccesible del terreno de depliegue, sólo resulta posible la distribución aleatoria de los nodos.

Esta tesis aborda tres problemas de posicionamiento de red. Los dos primeros problemas se han planteado de forma general, siendo de aplicación a cualquier tipo de escenario AmI. El objetivo es seleccionar las mejores posiciones para los nodos y mantener los nodos de la red conectados. Las opciones estudiadas son un posicionamiento determinista resuelto mediante el metaheurístico *Ant Colony Optimization* para dominios continuos, y un posicionamiento aleatorio, donde se realiza un despliegue *cuasi-controlado* mediante varios *clusters* de red. En cada cluster podemos determinar tanto el punto objetivo de despliegue, como la dispersión de los nodos alrededor de dicho punto. En este caso, el problema planteado tiene naturaleza estocástica y se resuelve descomponiendolo en fases de despliegue, una por cluster.

Finalmente, el tercer escenario de despliegue de red está estrechamente ligado al entorno AmI para la detección de caza furtiva. En este caso, utilizamos el método matemático de descenso sin derivadas. El objetivo consiste en maximizar la cobertura, minimizando a la vez el coste de despliegue. Debido a que los dos objetivos son opuestos, se utiliza un frente Pareto para que el diseñador seleccione un punto de operación.

Acknowledgements

First and foremost I would like to thank my thesis director, Javier Vales Alonso, for his excellent supervision. He has always been available to share his ideas and knowledge with me. Working with him has been a truly enriching and stimulating experience. I can therefore only express my sincere gratitude and appreciation for his considerable support, generosity and commitment to this work. I would also like to thank Joan García Haro for his selfless support and encouragement.

I would like to extend my gratitude to Juan José Alcaraz for contributing his knowledge about dynamic programming adapted to intelligent decision-making. I am also deeply grateful to the other co-authors of the articles on which this thesis is based. To each and every one of you, many thanks for your knowledge and support.

I would also like to extend a special thank you to all my colleagues in the telematics engineering department with whom I have been privileged to share some good moments. Thanks to you all for making this work so pleasant and enriching.

I dedicate this thesis to my parents María Dolores and José María to whom I owe so much, and to Concha and Marcos without whose affection and encouragement none of this would have been possible.

Financial support

Below are the main sources of funding for this thesis:

- Project CALM TEC2010-21405-C02, funded by the Spanish Ministerio de Ciencia e Innovacion (Ministry of Science and Innovation).
- "Programa de Ayudas a Grupos de Excelencia de la Región de Murcia", funded by Fundación Seneca, Agencia de Ciencia y Tecnología de la Región de Murcia (Plan Regional de Ciencia y Tecnología 2007/2010 / Regional Plan for Science and Technology 2007/2010).
- Project "Sistemas de inteligencia ambiental para asistencia a deportistas con perfiles específicos" (AISAS) DEP2006-56158-C03, funded by the Spanish Ministerio de Ciencia e Innovación (Ministry of Science and Innovation).
- Project "Sistema Autónomo para el Entrenamiento personalizado de atletas basado en Técnicas de Aprendizaje máquina e inteligencia Ambiental" (SAETA) 112/UPB 10/11, funded by the Spanish Consejo Superior de Deportes (National Sports Council).

Contents

List of figures

List of tables

List of abbreviations

ACK	*ACKnowledge character*
ACO	*Ant Colony Optimization*
ACO$_{\mathbb{R}}$	*Ant Colony Optimization for continuous domains*
AmI	*Ambient Intelligence*
ARQ	*Automatic Repeat reQuest*
B-MAC	*Berkeley Media Access Control*
CN	*Control Node*
CONOPT	*COde for large sparse dynamic NOnlinear OPTimization*
DP	*Dynamic Programming*
FDP	*Four-Directional Placement*
FFNN	*Feed-Forward Neural Networks*
GAMS	*General Algebraic Modeling System*
GIS	*Geographic Information System*
GMM	*Gaussian Mixture Model*
GPRS	*General Packet Radio Service*
GPS	*Global Positioning System*
HR	*Heart Rate*

IN	*Infrastructure Node*
ISTAG	*European Commission's Information Society Technologies Advisory Group*
k**-NN**	*k-Nearest-Neighbors*
LDP	*Level Discovery Protocol*
LROC	*Lunar Reconnaissance Orbiter Camera*
MDP	*Markov Decision Process*
MAC	*Medium Access Control*
MEMS	*Microelectromechanical system*
NASA	*National Aeronautics and Space Administration*
NFC	*Near Field Communication*
$\mathcal{NP}$**-hard**	*Non-deterministic Polynomial-time Hard*
RBS	*Reference Broadcast Synchronization*
RFID	*Radio-Frequency Identification*
RGG	*Random Geometric Graph*
SEPRONA	*SErvicio de PROtección a la NAturaleza*
SVM	*Support Vector Machine*
TDoA	*Time Difference of Arrival*
TSPN	*Timing-Sync Protocol for sensor Networks*
UE	*User Equipment*
UWB	*Ultra-Wideband*
WAC	*Wide-Angle Camera*
WLAN	*Wireless Area Network*
WSN	*Wireless Sensor Networks*

Chapter 1

Introduction

In 2001 the European Commission's Information Society Technologies Advisory Group (ISTAG) introduced the concept of *Ambient Intelligence* (AmI) [DBS+01]. ISTAG applies this new term to environments (home, work office, playground, shopping center, etc.) that are able to recognize and interact with the user in a *non-obstrusive* way and cooperate in undertaking tasks. In other words, such environments have a two-fold mission: (i) to not interfere with the activity being undertaken by the user and (ii) to enhance user skills (thereby facilitating performance of tasks). The concept of AmI is therefore associated with the capacity to understand a problem (in this case, the activity that involves the user) and to choose the best solution (often supported by previous experience). This explains the term *intelligent environments*.

Figure 1.1 gives a schematic view of an ambient intelligence environment. It shows three different parts or levels: physical, communication and logical. The first level is the physical layer and refers to the physical world where users undertake their activity. This level includes one or more users, the objects that surround them (environment) and that they perceive (with the senses).

The second level (the communication layer) main goal is to get the state of the physical world. Hence, it would consist of a set of devices, mainly with sensing and computation capabilities. This infrastructure is integrated in the objects in the environment in such a way that the user does not notice their presence. These devices are simultaneously able to monitor the environment: location of objects, ambient parameters (light, temperature, humidity, etc.), in addition to several user parameters (*e.g.* position, heart rate, etc.). These data represent the state of the environment and the user, which are continuously changing as they evolve with time. To become useful, these data must be collected and shared (*e.g.* for processing and interpretation). Therefore a certain amount of connectivity (usually wireless) must exist among the various devices integrating the communications network.

This network must capture the user's situation and environment and must also be able to transmit orders to act upon them or with them (using actuator devices). To generate these orders, the third level of Figure 1.1 (the logical layer) contains the tools for mathematical and logical analysis (classifiers, inference methods, and so forth) needed to process data captured by the sensors in the network layer. The aim of this process is to interpret the data captured by the communication layer (*i.e.* understanding the physical

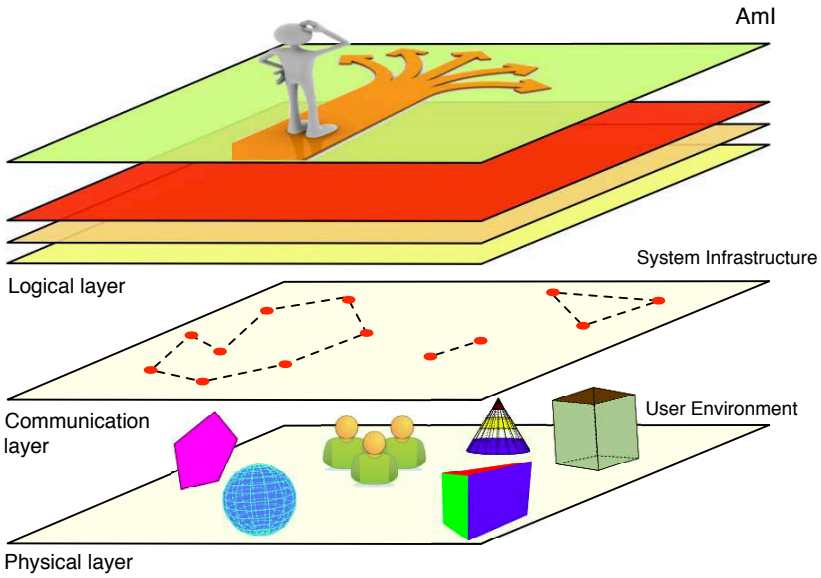


Figure 1.1: General structure of an AmI environment.

layer) and the generation of intelligent actions (*i.e.* decision making). In other words, the logical layer is capable of recognizing the situation of both the environment and the user, and, depending on them, generating the appropriate actions.

In short, the basic architecture of an AmI system involves equipping the physical world with intelligence. The communication layer is required to interconnect the physical and logical worlds.

In order to illustrate the way in which AmI environments can be beneficial for many everyday tasks, below we recount the *case of the traveler* ([DBS+01]).

> *Lola has just landed at the airport. When she travels to this country she carries very little luggage because she knows she can make use of AmI systems. Accordingly she only uses a "W-MovComm" computing and personal communication device, which she wears on her wrist. This enables her to pass through airport customs without requiring the customs officer to check her documents, since her visa has already been processed and her identity checked through the 'W-MovComm' as she walked through the airport.*
>
> *A car has been reserved and is stationed at a specific place in the car park. Lola does not need a key. The car opens when she approaches and the engine starts at the touch of a button. She drives the car to the conference hotel where she will be giving a paper the following day. The hotel is located in the city center. A guidance system indicates the best route avoiding traffic jams. Attempts have been made to solve the problem of traffic in the city center by restricting access. Lola has an entry permit because she has a parking reservation at the hotel. The cost of preferential access to the center is automatically negotiated between Lola's 'W-MovComm' and the systems at the car hire offices and the hotel. The cost of access is charged to Lola's*

business account.

Amanda, Lola's daughter, is at home. An AmI system at Lola's home has detected that Lola is in a place where she can hold a telephone conversation and tries to establish communication. But Lola prefers to wait until she arrives at her hotel and concentrate on driving the car.

When she arrives at the hotel, Lola is directed to a specific parking space. The hotel buttons is waiting to help her with her luggage. When she enters her room, it adapts to her personal preferences: temperature, intensity of light, music and video. She needs to work on her next day's presentation and take a bath. She adjusts the intensity of light using her voice and requests that a bath be prepared. Then she makes a video call to her daughter. Before going to bed, she works on her presentation and saves it on her company's servers. The next day, she goes to the conference hall where she can download her presentation. When Lola enters the hall, she raises the access thresholds to the communication system of the other delegates to block everything except emergency messages. When she finishes her presentation, Lola lowers the delegates' communications barriers.

The previous example shows several AmI systems inter-communicating and cooperating. For instance, a network of hundreds even thousands of sensors monitor traffic on the main roads into the city and in the inner city grid. This information enables the AmI system to provide customized recommendations about speed, route and lane. Another AmI system is installed at the home and comprises a network of devices that provide a surveillance service. This system could also manage the intelligent use of energy (heating, electricity, hot water, etc.) according to the needs of the house occupants. The other AmI environments included in the example are the airport and the hotel. In this case, we see how the user can concentrate on the objective of the journey (the presentation), while several AmI environments help her to carry out the other activities involved in traveling.

AmI environments embrace an enormous variety of applications. We can think of other scenarios for applying AmI environments, thus assisting the user in a multitude of everyday situations. Such scenarios may include applications where a user is not the direct beneficiary of the actions or decisions taken by the AmI environment. For example, in agriculture there are many crops (*e.g.* grapes and some fruit trees) whose cost and complexity of cultivation (numerous factors may affect harvests) require optimized production [Gra06]. For this purpose, we can deploy a network of sensors and actuators to manage vast areas of crops, thereby increasing productivity and improving harvest quality. In this case, the sensors could collect information about variations in the ground and cultivation environment (concentration of food substrate, light, altitude, humidity and temperature of the ground and atmosphere). In this example, the AmI system can take useful decisions, such as controlling the concentration of pesticides, fertilizer and water supplied. Sensing and monitoring can be undertaken on a very small scale to the level of detail required: depending on the specific needs of each area (or even plant), by controlling levels of fruit glucose, etc. Different application scenarios can be consulted in the following works [DBS+01], [D. 01] and [Sad11].

Another example, which has been closely scrutinized in this thesis, is the world of

sports. For example, a system that assists an athlete during training. A network of sensors can simultaneously monitor the athlete's performance and environmental conditions (heart rate, blood oxygen level, temperature, altitude, humidity, etc.). Bearing in mind the athlete's previous training activity (historical records), the system would be able to evaluate current performance data and indicate the best training options. Chapter 2 and Chapter 3 focus specifically on an AmI environment oriented towards sports training. Another

AmI environment application examined in Chapter 4 is a system capable of detecting and locating the origin of a gunshot. An acoustic sensor network captures the sound and records it against a time mark. A method based on hyperbolic multilateration leads to the location of the gunshot. Early detection of the gunshot is a key aspect of system design. This system has been devised to detect poaching in national parks, thereby demonstrating the possibilities of AmI systems in welfare activities not directly related to humans.

Implementing such heterogeneous AmI environments requires the promotion of many different fields of research. The next section attempts to identify some of the main research efforts in AmI environments.

1.1 Some research challenges of AmI systems

The scenarios presented above include aspects related to intelligence, communication and sensing/acting tasks, leading to some of the key challenges of AmI:

- **Adaptation to users:** User needs evolve with time and the system must react continuously to fulfill them. This requires coordinated responses with the user activity and with changes in the environment. Furthermore, since several users can use an AmI system simultaneously, users must be identified first. The identification process must be performed securely and accurately.

- **Real-time operation:** The system must operate in real-time according to user requirements. This requires both efficient processing and communications:

  - *Processing*. Decision-making centers are points in charge of analyzing and processing data to generate a response. Occasionally, service may also require collaboration (or negotiation) between several decision-making centers. Each step in the processing requires a time, which must be minimized in order to provide the user with a rapid response. Commonly the main contributor to operation time is the decision part, *e.g.* due to the complexity of the problem to be solved. Techniques for decision-making performing analysis in real-time must be developed.

  - *Communications*. Another critical part is the availability of fast and reliable communications to improve system response time. For example, by reducing unnecessary control transmissions. In this area, protocols for the different communications layers must be explored, from MAC (medium access control) to transport protocols.

- **Network deployment:** The quality of decisions (and therefore of the AmI system) depends directly on the quality of the information available. The network's sensor nodes within its area of sensing capture these data. In addition, data must be efficiently transported to decision-making points (as discussed above). Therefore, connectivity between nodes must also be assured. In this context, selecting node position is fundamental in order to cover interesting phenomena. Such phenomena occasionally need to be captured by more than one sensor (for example, to localize a sound source). In addition, the network is required to monitor large extensions leading to large complex networks.

  These special characteristics of the AmI systems make determining the positions of network nodes a daunting task, which must be addressed.

- **Human interfaces:** User interfaces must be intuitive and user-friendly. They must also adapt to user requirements (*i.e.* pervasive adaptation), thereby facilitating natural communication. Challenges in this area are wide-ranging. Among them are emotional and gestural detection, and new interfaces (*e.g.* Google glass project) with more sensing and communication capabilities.

- **Reconfiguration:** Once an AmI system has been made operative according to specific service needs it is difficult (or practically impossible) to modify. This is because the devices are integrated in objects in our environment, which cannot be repositioned or easily accessed. Over time, new needs may arise that call for operational change or the addition of new applications. AmI system components should be adapted to allow such changes. This requires a paradigm shift in application design. Until now each component was designed to provide a specific function and interaction with other components. In AmI systems, however, we must consider that each component has a basic capability. We could then organize some of these capabilities to construct new functions. Parts of an AmI system originally installed for one service could then be modified. Achieving such versatility in AmI systems requires appropriate software. Some requirements for software include being updatable, highly available and able to work with new hardware. In order to efficiently construct software for these systems, new development methods and tools are needed.

- **Technology of infrastructure:** The lines of work outlined above are related to the functionality of AmI systems (generating useful decisions for users). Further consideration should also be given to other aspects linked with technologies associated with the infrastructure. AmI system infrastructure is mainly composed of sensor devices seamlessly integrated in objects in the surrounding environment. For example, in buildings, electrical appliances, furniture, urban furniture, etc. Once installed, they are therefore difficult to access for replacement, reconfiguration, battery recharge and relocation. These features and functionality call for these devices to be small, energy-self-sufficient and equipped with suitable communication capabilities. The following actions are therefore required:

  - Develop and manufacture devices, increasing production efficiency to reduce

cost. This would broaden use, enabling these devices to be integrated in numerous objects in the environment.

- Promote technologies that provide energy efficiency. This requires developing batteries that can be adapted to small devices and that can store and retain a maximum charge for their size when not in use. Other systems integrated in devices (communication, control, etc.) must be designed to support intrinsic energy constraints. In this field, development of low-energy processors is vital.

  In addition to aspects of device manufacture, we can also focus on energy resource optimization by means of techniques associated with the operation of these networks. For example, techniques for *on-off* devices (alternating operation and quiescence) and redundancy (replacing a device with a depleted battery for another nearby). It is also necessary to research other sources of energy (*e.g.* green) that comply with the size, computation and communication requirements of AmI systems and their components.
- Research short-range low-power wireless communication (*i.e.* by reducing energy consumption). For example, communication techniques such as *ultra-wideband* (UWB) and *near field communication* (NFC) can be explored.

1.2 Thesis objectives

This thesis aims at advancing on the design and development of novel AmI systems in three main research lines:

- **The development of AmI systems to enable novel applications**. In particular, development of intelligence environments applied to sports and platforms for natural ambients protection. Chapter 2 and Chapter 4 respectively, present the architectures of these systems.

  The aim of the AmI system applied to sports is to assist athletes with their training. To that end, we developed a system composed of a *wireless sensor network* (WSN) over a cross-country circuit and mobile sensors worn by users to monitor their biometrics (see Chapter 2 and Chapter 3).

  Furthermore, for the protection of natural environments, we propose a system comprising a network of acoustic sensors whose aim is to locate gunshots to detect poaching. As with the previous case, a network of sensors must be deployed. In this case, aspects such as network deployment and synchronization are particularly important. Chapter 4 addresses these challenges.

- **Decision making in AmI systems**. Decisions will be based on information supplied by several elements within the system (WSN, user terminals, etc.). Specifically, our study focuses on the two AmI environments outlined in the previous point:

  (a) *In the training system*: The objective is to select in real-time training routines that have been personalized for each athlete. Until now, training systems have been incomplete because they only provided useful data at a later stage, that

is, once the training session was over. Furthermore, the monitored data was incomplete: only some biometric parameters of the athlete were recorded and environmental parameters were usually disregarded. In our study, both problems are addressed by using suitable decision-making systems. Chapter 3 discusses this topic.

(b) *In the natural environment protection system*: The aim is the early detection and location of gunshots. Poachers are chiefly arrested after they have repeatedly committed the offense (several times). It is therefore vital that offenders are caught just after collecting the game. In order to do this, the gunshot and its location must be detected as soon as possible. This poses a challenge because of the characteristics of the hunting scenario: the existence of obstacles, lack of visibility (even from a watchtower), size of the surveillance area (it could be a national park) and the difficulty of installing any kind of customary surveillance system (*e.g.* around the perimeter). In this study, we propose collecting time marks in order to locate its origin. For that purpose, we devise a plan to deploy a WSN and develop a novel energy-efficient mechanism that will enable all the nodes to be synchronized. Chapter 4 describes these mechanisms.

- **Optimal node placement**. As indicated previously, decision making depends on the quality and quantity of the available data. Improving quality of the data directly leads to better decisions in AmI environments. Planning deployment requires bearing several, occasionally conflicting, objectives in mind. Namely, improving connectivity and network coverage (understood as quantity and quality of the data captured) are two of the main interests. In a real scenario the importance of the captured data is not distributed uniformly over the deployment surface. Important areas must be covered, but it could be also necessary to deploy nodes in areas without interest to connect the network. This compromise between coverage and connectivity becomes even more complex if the number of nodes available is restricted. In this thesis, we examine two types of network deployment:

  (a) *Deterministic*. During planning, it is possible to select the exact operating position for each of the nodes of the network (discussed in Chapter 4).

  (b) *Random*. In this case, there is uncertainty about the final position position of the network nodes. These deployments are even more challenging than deterministic ones (discussed in Chapter 5).

1.3 Background

This section introduces the precedent works related to AmI applications, decision-making and deployment methodologies. The objective is to situate the innovations carried out in this thesis. First, the Section 1.3.1 inspect some previous works on sport training systems; the compiled works are scarce because of the novelty of this kind of AmI systems, especially when considering outdoor scenarios. Second, the Section 1.3.2 discusses some statistical and numerical analysis methods employed as decision-making techniques. These

techniques are used later to manage the future evolution of athletes in Chapter 3. Besides, Section 1.3.3 enumerates some node deployment optimization problems. Mosts of these studies consider the improvement of network coverage sensing and node connectivity. Finally, Section 1.3.4 includes the related works on sensing and locating acoustic sources, which are related to the natural environments protection system developed in Section 4.3.

1.3.1 Sport oriented training systems

The use of wearable computing is, nowadays, common in athletes' training. As described in [PSLB08], the use of chronometers, photocells, contact platforms, microphones, photo or video cameras, magnetic resonance and X-rays machines, movement sensors and, in summary, every sensing device for physical or chemical parameters, is nowadays common in athletes' training. Many of these devices are not exclusive for elite athletes, but available to the general public (*e.g.* heart rate monitors). They are used for training monitoring, by athletes themselves or by their coaches, to improve physical performance. In fact, the widespread use of commodity hardware (*e.g.* the Apple's iPhone) has also lowered the barrier to create training sports mobile applications [SLF+08].

To some extent, the availability of these devices is just the first step towards the appearance of contextual services and AmI environments. Now, sport monitoring systems introduced real-time data collection, as well as user localization. Communications capabilities are provided by standard wireless network protocols (IEEE 802.11, Bluetooth/Wibree, Zigbee, etc.). Either these networks or specific ones, (UWB, RFID, GPS, etc.) may support location [MFA07]. Future developments will aim at expanding the range of monitored data (environmental conditions, detailed user data), and, to provide useful actions and information based on them. WSNs [AJK07] represent one of the enabling technologies for that evolution.

Several context-aware applications for athletes training have already been introduced. The focus of these works is on collecting data related to specific applications. However, these data are not used directly to obtain real-time feedback, but they are externally analyzed by human experts. Next, works belonging to this group are described.

A system to supervise elite sports training is proposed in [BK06]. Their work shows how sensors expand available data to study exercise execution. Three use cases are considered: table tennis, biathlon and rowing. In the case of table tennis, the system detects impacts of the ball on the table, computing their positions by means of vibration triangulation, to display throw accuracy. In the biathlon case, a laser positioning system analyzes the motion of the rifle barrel before and after the shot. Finally, in the rowing case, the system calculates the effort that is applied to the oar.

In MarathonNet [PLB+06] a WSN monitors runners in marathon events. Sensors on runners collect data about heart rate, time and location. These data are sent via base stations along the track to a central database, where they are analyzed *off-line*. Base stations can communicate with the central database by means of GPRS, WLAN, or a wired network link. In the system developed in this thesis, the sensor nodes have similar functionality, but they also act as information routers.

The system in [MS05] assists professional skiers. Using accelerometers and force-sensing resistors, skiers can obtain data about their movements and visualize them, along

with video footage, once the exercise has finished.

SESAME [Hai06] proposes the combination of two data sources, video and sensed parameters, to improve the performance of novice and elite athletes. Major competitions, like the Olympic games, typically pioneer the adoption of new technologies. As an example, in [Chi05], the author introduces the SensorHogu systems, which embeds piezoelectric force sensors on body protectors, to recognize valid scoring kicks in Taekwondo. The commercial product Team^{2}Pro [Pol11] allows to record and study fitness data in real time for up to 28 players.

None of the previous works consider real-time feed-back. In other words, the human expert (*e.g.* the coach) carries this task. Some systems aim at simplifying this manual task are presented in [JTN$^+$, LLJ08, LJTL08]. They describe a system which monitors dynamic data from cyclists and their bicycles and provides *automatic* real-time feedback. For instance, the system can advise the group to change the formation, split, or to increase or decrease the speed. In [JTN$^+$] decisions consist of simple rules based on thresholds on monitored variables, whereas in [LLJ08, LJTL08] emphasis is put on the control algorithm taking into account the *heart rate* (HR) of the cyclist, the headwind, and other parameters.

Reference [SB08] introduces a sensor system which provides immediate feedback to alert users about incorrect movements and body positions in snowboarding. The prototype uses sensors attached to the human body and inserted into the boots. The system detects body position (*e.g.* knee bending) and calculates deviations from positions previously captured from experienced subjects. However, this development is unaware of the surrounding environment (*e.g.* slopes) to improve decisions.

Paper [AB08] presents a score system for golf swing motion. The golfer body motion is captured by a wireless network of inertial sensors which extracts snapshots of the orientation data of the user body in the golf swing. The orientation information is compared with correct motion rules, and a score of the exercise is computed.

Reference [GLGJ09] also develops a feedback system based on sensors which capture the movements of a golf swing. The swing motion is preprocessed locally and sent to a base station, *e.g.* a PDA, for further analysis. This system does not consider environment conditions and, thus, does not incorporate devices embedded in the environment. The quality of the swing motion is expressed as the amount of deviations from the target line, and is computed by linear discriminant analysis.

Finally, there are also a great deal of works that utilizes *dynamic programming* (DP) [BD62]. This method results specially useful in predicting future evolution of complex systems which involve several statistical variables. Dynamic programming techniques have been applied in many areas such as economics, medicine or artificial intelligence, which included some developments in sports. In [Cla88] the authors calculate at any stage of the innings the optimal scoring rate, including an estimate of the total number of runs to be scored and the chance of winning in an one-day game. The results obtained help to study optimal batting tactics (*e.g.*, the best run rate at any stage of the innings).

Optimization problems in outdoor sports like competition glider flying and sailboat racing are challenging due to uncertain environmental conditions. The pilot must take a continuous series of strategic decisions, with imperfect information about the conditions he will encounter along the course and later in the day. In a competition, these decisions

seek to maximize cross-country speed, and hence the final score in the contest; even in noncompeting flying the pilot must fly fast enough to complete the chosen course before the end of the day. The work in [AT04] addresses the problem of uncertain future atmospheric conditions by constructing a nonlinear Hamilton-Jacobi-Bellman equation for the optimal flight speed, with a free boundary describing the climb/cruise decision.

None of the above works provide *real-time feedback* to the athletes. Previous works use information of the athletes or in some cases from their close environment. AmI allows to expand the environmental characterization to include data from the whole training field. Including this information improves the quality of decisions in some sports (*e.g.* running) as we demonstrate in Chapter 3. As it is showed in the next section, the intelligent part of AmI sport oriented environment, developed in this thesis, aims at estimating *future* athlete performance, rather than only determine the current quality of the exercise. Therefore, feedback assists athletes in order to maintain their target activity, instead of modifying it when it is not correct. Another specific characteristic of the prototype presented in this thesis is that, during the training, it evaluates if the decisions were correct and incorporates this information to improve further feedback (see Chapter 3).

1.3.2 Decision making techniques

This section introduces the supervised learning techniques that we have used in this thesis (Chapter 3):

1.3.2.1 An introduction to machine learning

Generally speaking, learning is considered to be a natural skill, leading to a kind of adaptation to the environment. Engineering problems frequently uncover the need for some kind of adaptation for a correct functionality in the environment in which machines must operate. More specifically, programs must be created that are capable of generalizing behaviors based on unstructured information supplied by way of examples. It is therefore a process of knowledge induction. This task requires a set of measures or examples associated with the process that needs modeling. This kind of learning, known as inductive learning, becomes learning by example, which is fundamentally a problem of function approximation about which only a set of points is known.

There are two basic methods. The first, known as *unsupervised learning*, where the entire modeling process is carried out on a set of examples only composed of input data. There is no information about the categories of these examples.

The second type of learning, known as *supervised learning*, is based on a set of training data (input) for which we observe a series of results (output) for a group of objects (such as persons). These data are used to construct a prediction or apprentice model that will forecast the results that will be obtained for a new object. The quality of the apprentice is determined by the accuracy of these results.

This learning process is undertaken through training monitored by an external agent (supervisor, master), which determines the response that should be generated by the network from specific input. The supervisor checks the output generated by the system and,

in the event of it not coinciding with the expected result, will proceed to modify the structure of the decision method.

Three ways of implementation are usually distinguished for supervised learning:

1. Learning by error correction: it consists of adjusting the weights of the network connections depending on the difference between the desired values and those given in the output.

2. Learning by reinforcement: this kind of learning is slower than the former and is based on the idea of not being in possession of a complete example of the desired behavior; that is, of not indicating during training the exact output desired of the network in the face of specific input. Here the role of the supervisor is limited to indicating by means of a reinforcement signal whether the output obtained in the network meets that desired (success $= +1 = -1$). Depending on that outcome, the weights are adjusted according to a probability mechanism.

3. Stochastic learning: this basically consists in making random changes to the values of the weights and evaluating their effect on the desired object and probability distributions. A network that uses this type of learning is the Boltzman Machine Network, created by Hinton, Ackley and Sejnowski in 1984, and the Cauchy Machine Network, developed by Szu in 1987, ([HSA84] and [SH87], respectively).

Supervised learning techniques make common use of cross-validation statistical methods for testing. Next we explain validation methodology and supervised learning techniques used in this thesis:

- k-Nearest-Neighbors, and
- interpolation based on (m, s)-splines.

We deserve a separate section to dynamic programming techniques which is also studied for decision-making in AmI.

Validation Cross-validation is the statistical practice of splitting a data sample into subsets so that analysis is initially undertaken in one of them. The other subsets are kept for subsequent use for confirming and validating the initial analysis. This technique is frequently used in Artificial Intelligence to validate models generated from a data set or a sample.

Simple validation consists of dividing sample data into complementary sets, using one of them to construct the model (*training set*) and the other to measure the error ratio of the constructed model (*test set*).

Cross-validation applies 'k' folds to simple validation by dividing sample data in 'k' sets. A model will be constructed and evaluated at each iteration by using one of the sets as a test set and the rest as a training set. In the end, by calculating the arithmetical mean of the error ratios obtained, we will reach the error ratio for the final sample.

The objective of cross-validation consists in estimating the expected level of adjustment of a model for a data set that bears no relation to the data used for training the model.

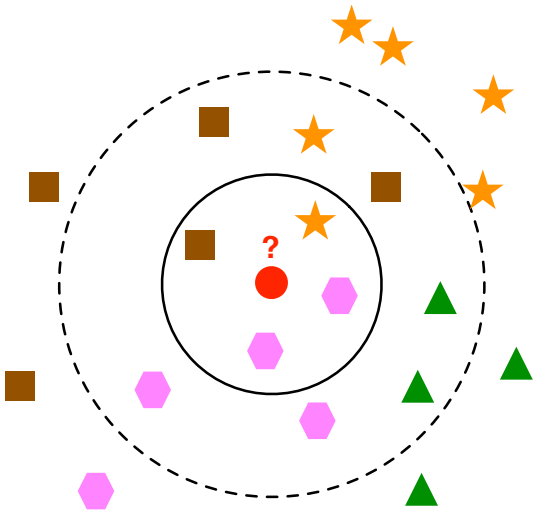


Figure 1.2: An illustration of the k-NN classification method.

It can be used to estimate any quantitative measure of adjustment suitable for the data and model. For example, let us suppose that we have a model with one or more unknown parameters and a satisfactory model for the data set (training data). The adaptation or optimization process involves adjusting the model to the training data in the best possible way. If we take a sample of data validation, from the same population as the training data, the model does not generally adjust to the validation data as it does to the the training data. This is called *overfitting* and in all likelihood occurs when the size of the set of training data is small or when the number of parameters in the model is big.

k-Nearest-Neighbors k-NN [FH89] is a supervised classification method used to estimate the density function $f(x/C_j)$ of input variables x for each class y_j.

This classification method estimates the value of the probability density function or directly the a posteriori probability that an x element may belong to class C_j, based on information provided by the set of training samples. In the learning process no supposition is made about the distribution of the predictor variables.

In pattern recognition, the k-NN algorithm is used as a method for classifying objects (elements), based on training by examples within the space of the elements. That is, the class of an object will be determined by the class of objects (neighbors) nearest its position (see Figure 1.2).

- *Training algorithm.* Training examples are vectors in a multidimensional feature space. Each example is described in terms of p attributes, considering q classes for the classification. The values of the attributes in the i-th example (where $1 \leq i \leq n$) are represented by the p-dimensional vector $x_i = (x_{1i}, x_{2i}, \ldots, x_{pi}) \in X$.

  Space is partitioned in regions according to the locations and labels of training examples. A point in space is assigned to class $\mathcal{C}$ if this is the most common class among the closest k training examples. Euclidean distance is generally used:

$$d(x_i, x_l) = \sqrt{\sum_{r=1}^{p} (x_{ir} - x_{lr})^2}$$

The training phase of the algorithm consists in storing the characteristic vectors and labels of the classes of training examples. In the classification phase, evaluation of the example (whose class is unknown) is represented by a vector in the feature space. The distance between the stored vectors and the new vector is calculated, and the closest 'k' examples are selected. The new example is classified with the most common class of the chosen vectors.

This method assumes that nearest neighbors give us a better classification and is undertaken by using all the attributes. The problem with this assumption is that many irrelevant attributes may dominate the classification: *e.g.* two relevant attributes would lose weight among twenty irrelevant ones [FHT01].

To correct this possible bias, a weight can be assigned to the distances of each attribute, thus giving more importance to the more relevant attributes. Another possibility is to attempt to determine or adjust the weights with known training examples. Finally, before assigning weights, it is advisable to identify and eliminate attributes considered to be irrelevant.

- *Classification algorithm.* If we select $x^\star$ to be classified, $V = \{x_1, \ldots, x_k\}$ being the set of the k nearest neighbors to $x^\star$ in the learning examples, the regression is

$$\hat{f}(x^\star) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where $\delta(a, b) = 1$ if $a = b$, and 0 in any other case.

The $\hat{f}(x^\star)$ value returned by the algorithm as an estimator of $f(x^\star)$ is only the most common value of f between the k nearest neighbors to $x^\star$. If we choose $k = 1$, then the nearest neighbor to $x^\star$ determines its value.

- *k selection.* The best choice of k essentially depends on the data. Larger values of k make the algorithm less sensitive to *noise* [CD07, MRS08] on the classification, but make boundaries between classes less distinct. A good k can be selected through optimization. The special case in which the class is preselected to be the closest class to the training example (when $k = 1$) is termed *nearest neighbor algorithm* [FHT01].

  The accuracy of this algorithm can be severely degraded by the presence of noise or irrelevant features, or if the feature scales are not consistent with what is considered important. Considerable research effort has gone into selecting and scaling features to improve classification. An approach is to scale features according to the mutual information of the training data with the training classes.

- *Closest neighbors with a weighted distance.* The contribution of each neighbor can be weighted according to its distance from $x^\star$, the example being classified, by

assigning more weight to nearest neighbors [FHT01]. For example, we can weight the vote of each neighbor according to the inverse square of the distances:

$$\hat{f}(x^\star) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

where $w_i \equiv \frac{1}{d(x^\star, x_i)^2}$.

In this way, there is no risk of allowing all the training examples to contribute to the classification of $x^\star$, since their distance prevents them from having associated weight. The disadvantage of considering all the examples is the slow response. It is advisable to use a local method which considers all the nearest neighbors.

This improvement is highly effective for many practical problems. It is robust against data noise and sufficiently effective for large data sets. By taking weighted averages of the k nearest neighbors, the algorithm can prevent the impact of examples with isolated noise.

Splines of (m, s) order Numerical approximation techniques based on splines are extensively applied in engineering areas such as signal processing [Uns99] and surface fitting area [LS86]. Among the different spline techniques, we selected (m, s)-splines [LdSA89] for their favorable characteristics to our research: they allow to face multivariable problems, the problem domain is not required to be a mesh grid (that is, data used to compute the approximation function can be at any point in space), and the computational load is low.

The theory of (m, s)-degree splines derives from the study of the *semi-Hilbert* space of functions, $X^{m,s}$, as is described in [LdSA89]. Let $n \in \mathbb{N}^*$ be the number of dimensions considered. Let $m \in \mathbb{N}^*$ and $s \in \mathbb{R}$ be parameters such that $(n/2 - m < s < n/2)$ holds. Given a set of elements $A \in \mathbb{R}^n$, and its corresponding values $\beta \in \mathbb{R}$ at those locations, the goal is to determine a mapping σ_ε as smooth as possible that approximates the observed data. The problem can be formulated as the minimization of the penalized sum of squares:

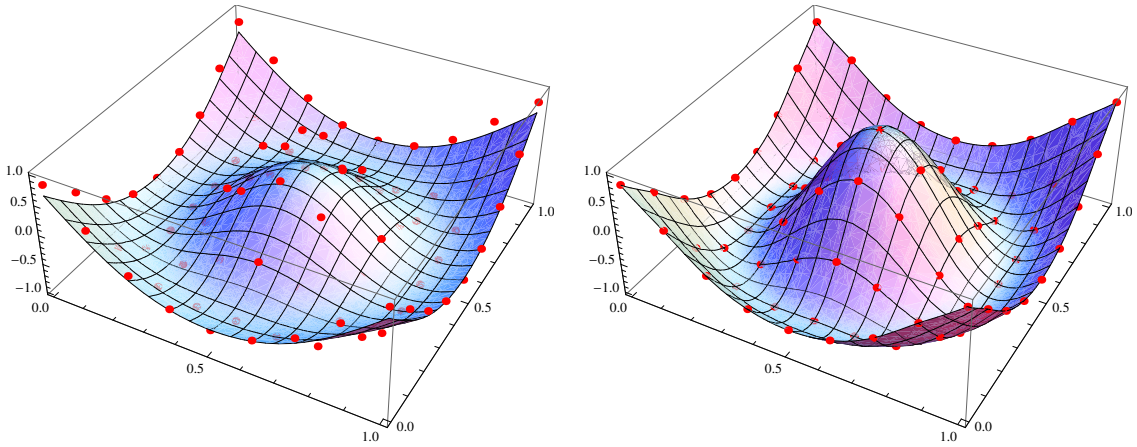
$$J_\varepsilon(v) = \left|\rho v - \beta\right|^2 + \varepsilon \left| v\right|^2_{m,s}$$

where $\varepsilon > 0$, $v \in X^{m,s}$, and ρ is an operator defined as

$$\rho v = (v(a))_{a \in A}$$

Value $\varepsilon \left|v\right|^2_{m,s}$ represents a smoothness penalty on function v. The *semi-norm* $|v|_{m,s}$ is specifically defined in [LdSA89, LdSPPT01]. Then, the *approximation spline of degree* (m, s) relative to A, β, and ε, is a function σ_ε such that $\forall v \in X^{m,s}, J_\varepsilon(\sigma_\varepsilon) \leq J_\varepsilon(v)$. Considering $\mathcal{P}_m$ as the ring of real polynomials of n variables, $x_1, x_2, \ldots, x_n$, and degree $\leq m$, namely

$$\left\{ p \in \mathcal{P}_m, \quad p(x) = \sum_{\alpha_1 + \cdots + \alpha_n \leq m} x_1^{\alpha_1} \cdots x_n^{\alpha_n} \; ; \; x \in \mathbb{R}^n \right\}$$



(a) Approximation ($\varepsilon = 0.005$) (b) Interpolation

Figure 1.3: Spline functions examples ($m = 2, s = 1/2$).

In [LdSA89] it is demonstrated that σ_ε is unique and it can be obtained by solving the following system of equations:

$$\begin{cases} \varepsilon \mathbf{C}^* \lambda_b + \sum_{a \in A} \lambda_a K_{2m+2s-n}(b-a) + \sum_{j=1}^{M} C_j p_j(b) = \beta_b \,, & \varepsilon > 0 \\ \sum_{a \in A} \lambda_a p_j(a) = 0 \,, & \forall j = 1, \ldots, M \end{cases}$$

where $\{p_j, 1 \leq j \leq M\}$ is a space basis of $\mathcal{P}_{m-1}$, $M = \dim \mathcal{P}_{m-1}$, and $\lambda_a, a \in A, \lambda_b, b \in A$, and $C_j, 1 \leq j \leq M$ are coefficients to be determined. The function $K_{2m+2s-n}(x)$ is

$$K_{2m+2s-n}(x) = \begin{cases} |x|^{2m+2s-n} & , \text{ if } 2m+2s \neq 2l, l \in \mathbb{N}^* \\ |x|^{2m+2s-n} \log|x| & , \text{ if } 2m+2s = 2l, l \in \mathbb{N}^* \end{cases}$$

The constant C^* has the following expression:

$$C^* = \begin{cases} \frac{(2\pi)^{2m}}{C_1} & , \text{ if } 2m+2s \neq 2l, l \in \mathbb{N}^* \\ \frac{(2\pi)^{2m}}{C_2} & , \text{ if } 2m+2s = 2l, l \in \mathbb{N}^* \end{cases}$$

with

$$C_1 = \frac{\pi^{(2m+2s-n/2)} \Gamma\left(\frac{n}{2} - m - s\right)}{\Gamma(m+s)} \quad , \quad C_2 = \frac{2\pi^{(2m+2s-n/2)} (-1)^{(m+s-n/2+1)}}{\Gamma(m+s)\left(m+s-\frac{n}{2}\right)!}$$

Therefore, approximation (m, s)-splines has a coefficient matrix whose diagonal is $\varepsilon \mathbf{C}^*$. Moreover, selecting ε = 0 transforms the approximation function σ in an interpolation function (no error is allowed at the known points). Figure 1.3 shows examples of both surface approximation an interpolation from a sample data set.

1.3.2.2 Dynamic programming algorithms

Markov decision processes (MDP) allow us to expand the horizon of decision making. We have decided to carry out a study of these methodologies based on the premise of not

limiting the decision to be the best for the immediately subsequent action but to evaluate the future costs of decision making.

Markov decision processes In the theory of probability in statistics, a Markov process, named after the Russian mathematician Andrei Markov (*b* Ryazan, Russia, 1856; *d* Petrograd, USSR, 1922), is a random phenomenon dependent upon time for which a specific property is fulfilled: the Markov property. Commonly, a stochastic process with the Markov property or without memory is one for which the conditional probability over the present, future and past state of the system is independent.

Our objective is to design the operating rules of a system for best possible performance. In the field of discrete events systems, this means that, at each stage of the system, it is possible to act on a variable capable of influencing the future development of the system, thus enabling a certain degree of control over performance. If the system is modeled as a Markov chain, the control variable, generically called u, will determine the probabilities of the transition matrix. We will therefore have a $P(u)$ function, which maps a transition matrix for each u value. After each transition, the value for u in the new state must be decided. These systems are called *Markov decision processes* (MDPs). In MDPs, states (or transitions) must be associated with a specific cost or benefit. The problem addressed in an MDP is finding the best possible value of u (maximum benefit or minimum cost) for each state of the system, which is called an optimal policy. The greatest difficulty lies in optimizing both the expected performance in the state in which we take the decision, as well as the expected performance in future states we will reach as a result of the decision taken. The basis of the techniques that will enable us to find the optimal policy is *dynamic programming* (DP).

Dynamic programming In this context, 'optimizing' is equivalent to selecting (finding) the best solution from many possible alternatives. This optimization process can be seen as a sequence of decisions that provide us with the correct solution. Bellman's principle of optimality is often met: given an optimal sequence of decisions, all ensuing decisions will in turn be optimal. In this case it is still possible to continue taking basic decisions, in the knowledge that the combination will continue to be optimal. However, it will then be necessary to explore many sequences of decisions in order to reach the correct one, which is where dynamic programming comes in. Contemplating a problem such as a sequence of decisions is equivalent to dividing it into smaller subproblems, in principle more easily resolvable.

The principle of optimality basically means that the minimum cost $J_N(x)$ for N stages can be calculated by using the same recursion; that is, by beginning with the final cost $J_0(x)$ and calculating the minimum cost in the absence of stage $J_1(x)$ (see Figure 1.4),

$$J_1(x) = \min_u \mathbb{E}[g(x,u) + J_0(f(x,u))]$$

and so on and so forth, giving rise to the following recursive equation:

$$J_k(x) = \min_u \mathbb{E}[g(x,u) + J_{k-1}(f(x,u))]\,,\ k = 1, 2, \ldots, N$$

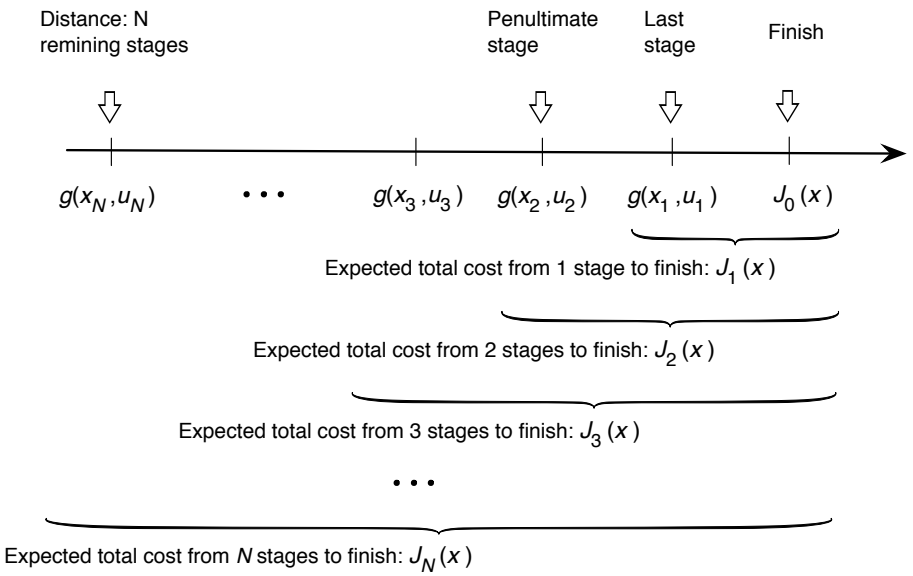


Figure 1.4: Recurrence of last stages cost.

Chapter 3 applies DP algorithms for decision-making in AmI system oriented to sports training.

1.3.3 Node placement strategies

This section collects related literature on network deployment focused on either deterministic or random scenarios. The problems addressed are related to both communications connectivity and sensing coverage, which have been extensively investigated in the WSN field. Most of the studies to date have aimed to optimize one or more key configuration factors to improve tasks related to the network being analyzed. Problems of this type were first introduced in [Gag92] and are denominated as *blanket coverage*. Blanket coverage related problems understand the quality of the WSN as a coverage metric. Other approaches (*e.g.* [AYY07, HLS+07]) have considered intrinsic or operative parameters, such as extending network lifetime or improving data fidelity. Commonly, these studies cope with node placement problems by means of optimizing some objective function. Optimization of deterministic deployments are surveyed in some works such as [YA08] and [GD08].

Let us focus the attention, first, on structured or deterministic deployment, *i.e.* where nodes can be installed in exact positions selected; second, a collection on related random deployment (*i.e.*, a *priori* the exact location of nodes is unknown) is presented.

1.3.3.1 Deterministic deployment

In [AYY07] the authors study the optimal positioning of WSN sink nodes under several performance metrics, such as lifetime and responsiveness of data delivery. This work studies two positioning techniques: static and dynamic. Static methods optimally place the sink at network start up phase and do not change during the operation time. Never-

theless, conditions may change during operation. For example, traffic load may change producing an unbalance distribution among nodes or network resources may change due to battery depletion. In these cases dynamic methods can propose a change of the sink situation to improve the performance of the network.

Since sensors mainly consume energy in data transmissions there are some works which try to extend lifetime by positioning nodes efficiently. In the paper [CCZ05] authors study the effect of node density on network lifetime, considering the network operative until the first node dies. They argue that deployment cost is proportional to the number of sensors and define an optimization problem with the objective of identifying the least number of sensors and their positions so that the network stays operational for the longest time. A clustering algorithm is presented in [HLS+07] to minimize the global energy consumption.

The work [DKK03] proposes a scenario in which a certain amount of data has to be captured by the network. The data is non-uniformly distributed and the authors propose a heuristic that relocates nodes in order to balance the load of data transmitted in zones with greater density of data, and removes sensors in zones with less density of data. The authors in [KSG08] propose a combinatorial optimization problem to maximize mutual information.

The work [WXTH06] presents an optimization model to identify the sensor locations that maximize the average probability of event detection per point. The search space is limited to the positions represented by a grid. This work attempts to achieve maximum importance, but restrict the positions of candidate sensors. The work [DC03] also limits the space for possible node placements to a grid. The authors present an iterative heuristic to achieve a given coverage goal using a minimum number of sensors. At each iteration, a sensor is placed at the best grid point. The algorithm terminates when the coverage goal is met, or when the bound on the sensor number is reached.

Finally, some recent works have included conditions to avoid isolated nodes in their connectivity models [VACRBD+08, CDWX08, AY08, AABP11, ZBT+10]. A model developed in this thesis also includes a condition to avoid isolated nodes.

1.3.3.2 Random deployment

Random placement is a classic problem which was studied long ago for air warfare [Lau57], yet it is still an active area of research. Many related works such as [Gil61, AD08, IKD04, LT04] focus on the topology of ad-hoc networks, which are usually modeled by means of *random geometric graphs* [PP03] (RGGs). In RGGs, vertices (*i.e.* nodes) are connected by an edge (*i.e.* link) if distance between them is below some limit (*i.e.* communication range). These works consider that in a network it is mandatory to obtain the largest connected component, and they study the conditions under which the *percolation* phenomenon occurs. This refers to the critical point in the average number of connections per node for which most nodes belong to same component (giant component), leading to a fully connected network.

Gilbert [Gil61] firstly studied this issue to find the critical density of a Poisson process at which a network can provide long-distance multi-hop communications. Authors of [AD08] analyze the degree of sensing coverage and connectivity of a WSN, assuming

that sensors are deployed following a Poisson point distribution. They discuss how node density affects network connectivity. The main finding of [AD08] was the establishment of a relationship between coverage and connectivity for this kind of network deployment. Percolation and random geometric graph theory are also utilized in [IKD04] to investigate the density of nodes required to guarantee coverage and connectivity in an uniform deployment of sensors. The authors present two incremental deployment algorithms which try to improve the placement of sensors in a series of consecutive deployments. The work [LT04] characterizes, by means of simulations, the fundamental coverage properties of a WSN. All these works aim at selecting basic network properties such as node density, communication range, and so forth, and their results are applicable to large scale *non-clustered* networks. In our work we also investigate connectivity and coverage, but we focus on clustered Gaussian deployments, rather than on uniform or Poisson point processes. Gaussian clusters are the realistic models for airborne deployments. As will be shown in the Chapter 5, unlike previous works, our aim is not to find conditions for percolation (full–connectivity of the network) but to maximize the sensing coverage of the cluster (which may occur in a sub-critical connection regime).

Some other works have also considered random Gaussian positions for the nodes. For example, reference [LRS09] considers a gridded area where a cell is considered to be covered if a sensor falls inside this cell. Like our work, sensor position randomness is modeled with a Gaussian distribution. The authors develop a deployment strategy such that a certain degree of coverage is met and the total number of deployed nodes is minimized. The results suggest that the best alternative for deployment is to aim the sensors at the center of each cell. However, this work does not address the general connectivity problem in the network, and uses a simplified coverage model. Authors of [ZC03] present two optimization algorithms which try to cover the vertices of a gridded surface with the minimum number of sensors. Similarly to our work, they consider a Gaussian model to describe non-deterministic placement but they do not consider the connectivity of the network. The work [WXA08] introduces two algorithms for Gaussian deployments which reduce the number of sensors while satisfying the required coverage and lifetime conditions. The authors use results from uniform random deployments to develop the connectivity model. As a drawback, their model can be applied only in scenarios of reduced dimensions with a high density of nodes. [IA04] investigates three functions to distribute sensor nodes over a surface to determine their respective fault tolerance properties. The comparison is performed with Gaussian, Uniform, and R-random distributions. In R-random distributions nodes are scattered uniformly with respect to radial and angular directions. Simulation results show that Gaussian and R-random placement strategies outperform the uniform patterned strategy because they tend to concentrate more nodes close to the sink. Besides, some recent studies [BJB08, XHTW10, BCR+11] have analyzed topologies formed by two-tier hierarchical clusters, which improve network lifetime and cover large surfaces easily. All these works develop and analyze the performance of different placement heuristics. By contrast, in our approach we develop an *optimal* stochastic program from the network model which allows us to directly compute the best cluster configurations. In Section 5.5 we show how our solution outperforms a reference heuristic.

In addition, different works have also proposed optimization problems for placement

problems, but they are more limited than our approach and do not focus on either random Gaussian clusters or on joint sensing coverage maximization. For example, [KSG08] proposes a combinatorial optimization problem to maximize mutual information. [WXTH06] presents an optimization model to identify the sensor locations that maximize the average probability of event detection per point. In this case, the search space is limited to the positions represented by a grid. This work is similar to ours because it attempts to retrieve maximum importance, but in our study we do not restrict the positions of candidate sensors.

Finally, most of the previous approaches do not analyze the impact of scenarios with non-homogeneous importance. The paper [HT05] mentions this issue and proposes an algorithm for covering important areas with more sensors. Research [PS04] provides a self-deployment strategy for robots which autonomously move to better positions. Dhillon *et. al.* Authors of [DCI02] discuss the inherent uncertainty of sensor readings and assign a probability measure to sensor detections. These probabilities help to model terrain effects. The authors also describe an algorithm for placing the sensors into a grid in order to provide adequate coverage. [AABP10] introduces a *multi-objective deployment algorithm* (MODA) which takes into consideration the following constraints: deployment cost (number of sensors), event detection probability, connectivity, and energy consumption. The proposed algorithm minimizes energy consumption and the number of sensors required to accomplish coverage goal. Similarly to our paper, the deployment scenario has non-homogeneous sensing requirements. The authors of [DKK03] propose a scenario in which a certain amount of data has to be captured by the network. The data is non-uniformly distributed and the authors propose a heuristic that relocates nodes in order to balance the load of data transmitted from zones with greater density of data, and removes sensors in zones with less density of data. The paper [IKD04] modeled coverage with a weighting function, which assigns a reward to each point covered that is inversely proportional to the distance from the sensor. Sensor positions that are likely to increase the total weight are preferred. None of these works address the placement problem from the perspective of optimizing clustered Gaussian networks.

Summarizing, previous works in random deployments have largely ignored both the clustered nature of WSNs and the non-homogeneous distribution of *importance*. This thesis is the first to develop an integrated model comprising coverage and network connectivity of Gaussian clusters. Chapter 5 describe our optimization problem from a theoretical perspective and provides analytical tools to solve it.

1.3.4 Location of acoustic sources

There exist some commercial solutions for acoustic signals detection (*e.g.*, gunshot sound), although they do not locate sound sources [Sil]. There also exist sound location tools for the military, but they are too costly and they cannot be deployed in large numbers in civilian applications [Def06].

Assuming a group of sensors has correctly detected the acoustic signal (*e.g.* by using *Gaussian mixture models* (GMMs) [CER05] [RLA06]), its location results from the combination of sensed data. A sensor only knows its own position and local time, an therefore these measures must suffice to locate the source.

In the literature there are several location estimators, such as triangulation [NN03] and trilateration [DSO08] methods. In the triangulation schema, every sensor determines the direction from which the acoustic event is detected, and then the location of that event is calculated as the intersection of the detection directions. Nevertheless, determining the direction of the acoustic event would make the hardware design too complex and costly, and the solution would be highly sensitive to terrain shape. A trilateration schema determines the location of the event from the distance between the source of the acoustic event and a fixed sensor. These distances can be obtained if the generation time of the event is known. However, sometimes, only the detection time is available to the nodes.

A directly observable acoustic signal between a couple of microphones is the *time difference of arrival* (TDoA) [KW07, HB04, Kri05]. The TDoA technique exploits the relationship between distance and transmission time when the propagation speed is known. Once the time delays are calculated, they are processed in order to estimate the location of the source [BAS97, VGT+07, CYE+03].

The hyperbolic location method [Pat05] consists in the minimization of an error measure that is a nonlinear function of the potential source location. This approach is scalable, since location accuracy increases with the number of nodes that detect the gunshot (see Section 4.3.5). Its domain can be a plane or a three-dimensional space.

1.4 Thesis structure and contributions

The rest of this thesis is organized as follows:

- Chapter 2 introduces the system architecture and the implementation of an ambient intelligence assistant for sports field. The network infrastructure and protocol adaptation for correct system operation are described in depth. Results of Chapter 2 have been published on the paper [VALMGC+10].

- Chapter 3 studies two broad types of decision-making procedures in order to control athlete behavior: (i) optimal classifiers, and (ii) dynamic programming. In the former part we analyze splines of (m, s) order and k-Nearest-Neighbors (k-NN). With these two methods we investigate the problem of selecting the best training decisions for the next stage. With the dynamic programming approach we consider the optimal decision for the whole training period. Adaptation of these three decision methodologies to our training AmI system is new. Moreover, the application of (m, s) splines to human biometrics classification problems is also a novel contribution. Study of the influence of environment variables (*e.g.* temperature) in decision-making process is new as well. This chapter is based on papers [LMVA09, VALMGC+10, LMVAGCn+10, VALMAGH11, VALMA+12].

- In Chapter 4 we solve two deterministic deployment problem. In these studies a non-homogeneous importance over the target area is considered. This models more precisely a real deployment scenario in an AmI environment. This feature of the models is a novel contribution, not previously addressed in the related works as discussed in Section 1.3. Besides, the goal of both models is to find the placement maximizing the captured importance, while addressing specific problem constraints.

On the one hand, the first model establishes connectivity constraints. Namely, to avoid isolated nodes (since their information is then lost). This first model can be used in any AmI outdoor placement problem due to its generality. It is solved by means of an adaptation of the *ant colony optimization* (ACO) metaheuristic applied to continuous domains. This part of the Chapter 4 is based on papers [LMVA12a, LMVA12b]

On the other hand, the objective of the model is also to minimize the installation cost of the network. Since both goals, maximizing the importance captured and minimizing the installation costs, are contrary, the solution is expressed as a Pareto frontier. This frontier is computed by means of a derivate-free descent method. This approach is used specifically to determine the best position for the acoustic sensors of the poaching AmI system. This part of the Chapter 4 is based on paper [GCACM+09].

Finally, also in Chapter 4 the design and development of the AmI system for ambient control is carried out. It includes the components required to detect and locate gunshots by means of an acoustic sensor network. A new algorithm is proposed for time synchronization of nodes which is specifically suited for real-time applications. This part is based on paper [GCACM+09].

- The final part, Chapter 5 is devoted to discuss and solve the problem of random clustered deployments. As in the previous deterministic problems the goal is maximizing the importance captured, subject to connectivity constraints. In this case, the exact position for each node can not be deterministically set, but has a random nature. The model assumes that clusters can be aimed to selected locations, and that its dispersion can be controlled. Therefore the goal is to select both variables for each cluster. Moreover, to cover large surfaces our model considers several clusters per scenario. This problem is addressed by decomposing it in several steps, one per cluster. Results of this chapter can be applied to any AmI network deployment of random nature. This chapter is based on paper [VAPGLMA12].

- Finally, Chapter 6 summarizes the main results of this thesis and points out some future research directions on the design and deployment of AmI systems.

Chapter 2

AmI platform for sport scenarios

This chapter introduces the system architecture and implementation of an ambient intelligence assistant for sport scenarios. The system is composed of a *wireless sensor network* (WSN) deployed over an outdoor athletic circuit which may monitor ambient variables, and by mobile elements worn by the users and which monitor their biometry, for example, their *heart rate* (HR).

We discuss in depth the architecture of this AmI system, as well as the protocols enabling data communication. Thus, we focus on the key aspects a designer of an AmI system must make: hardware, communications protocols, network topology, elements localization and so forth. This chapter therefore solves the sensing and acting parts of the general AmI scheme presented in the introductory chapter. This architecture serves as the basis for the decision-making procedures, which are addressed in the next chapter.

2.1 Introduction

As has been aforementioned in Chapter 1 WSN processing capabilities are growing rapidly, and they are getting increasingly embedded and seamlessly integrated in the physical world. WSN-assisted environments can be sensitive and responsive to the presence of people, allowing the development of *Ambient Intelligence* (AmI) services.

Among the activities where these technologies can be applied, sports may be one of the most benefited. For instance, in outdoor sports such as cross country running and jogging, practitioners commonly use wearable computing devices, which can provide useful telemetry about runner biometrics and practice-related events [BKP+10]. Some of the potentially useful data that can be captured by the system are the HR, track routes, speeds and distances, and so forth. Often, the complexity of the data collected requires subsequent analysis by a human coach, sometimes with the aid of specific software.

In common training systems, the performance of the athletes is only evaluated at the end of the training session, and sensed data are usually incomplete because generally only human biometrics are analyzed. In consequence, it is not possible to make real-time decisions during the training session, (*e.g.* to change the running pace or the track route) allowing the athlete to better accomplish their training objectives under variable conditions. Moreover, nowadays most training systems are limited because they do not take

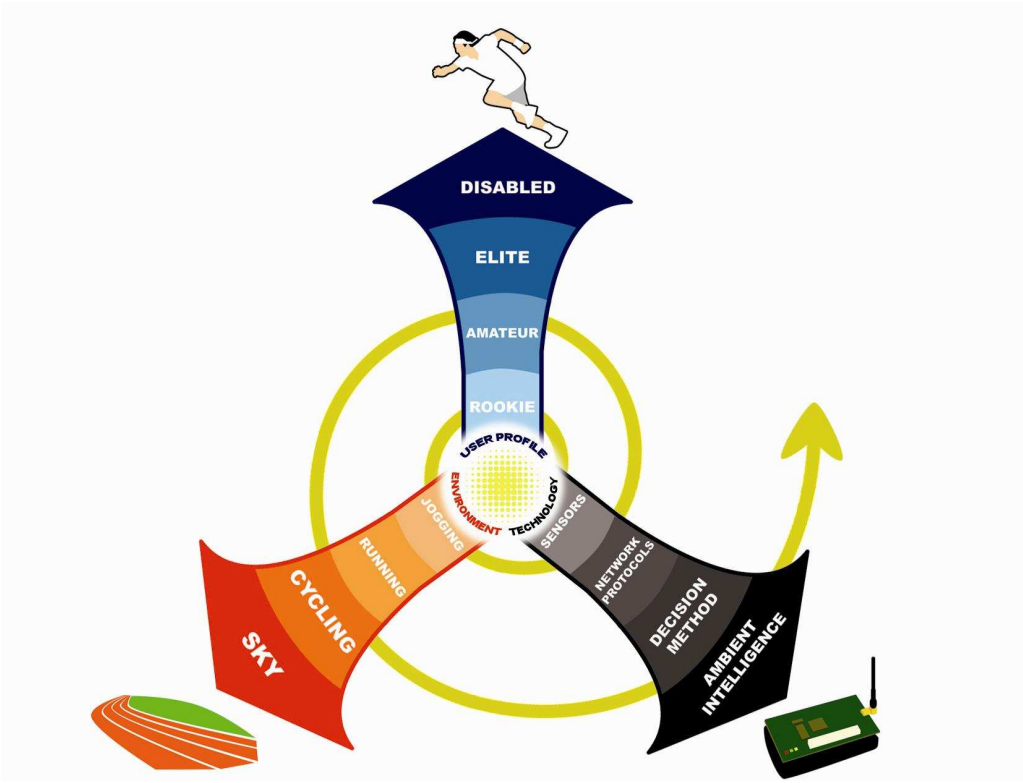


Figure 2.1: General vision of ambient intelligence aimed at sports.

into account the direct influence on athletes performance of the environmental conditions (*e.g.* temperature or humidity).

This thesis pursues to construct a system able to provide personalized assistance to athletes in their training. As shown in Figure 2.1, there are three driving aspects in this project: athletes' profile, environment, and computing and communications/decision technologies. Technology will provide adaptive coordination between the user and the environment, adapting the behavior of the system as it responds to changes in the environment or the user conditions, such as weather or athletes location. Since training conditions can rapidly vary, the system will be expected to make real time decisions to meet the user needs at their full potential. Thus, a decision engine, as well as efficient communications and localization protocols, are two key areas in the development of this training system. This chapter focuses on developing a prototype AmI system ready to perform test evaluations, and leaves the analysis of decision engines and the validation experiments of the prototype to next Chapter.

2.2 Outline of this chapter

The rest of this chapter is organized as follows. Section 2.3 discusses network design alternatives and justify the chosen network architecture presented along this chapter. Section 2.4 presents of the main infrastructure of the system and its functionality. Section 2.5

provides insights into network operation by deciding topology, communication protocols and data flows. Section 2.6 describes the hardware used in network implementation and software adaptation (*i.e.*, parameters tuning) for correct system operation.

2.3 Design preliminaries

Out goal is designing a general architecture to support athlete and environmental monitoring, in order to provide real-time feedback for training improvement. The infrastructure is mainly oriented towards open-field sports, such as running, cycling or skying. In these scenarios, athlete performance depends not only on her/his physical conditions, but also on terrain conditions (slope, temperature, wind, etc.). Environmental sensing is less critical for indoor sports and athlete sensing becomes highly specialized, as described in Section 1.3.1. Therefore, our approach focus only on outdoor scenarios.

Two types of data sets are important to characterize user performance:

- A *static* set, which has information that does not change along training, from the environment (*e.g.* terrain slopes, intrinsic exercise, pathway configuration, sensing nodes position, etc.) and from the user (*e.g.* performance profile, age, skill, training goals, etc.).
- A *real-time* set, with updated environment information (temperature, wind, visibility, etc.) and athletes' data (heart rate, body movement, elapsed training time, etc.).

A control element processes these data and issues commands to direct user exercise. Two approaches are possible for the implementation:

- A user-centric implementation, where the user equipment has the sensing devices necessary to monitoring data from the environment and from the athlete. This terminal is also in charge of taking the decisions to assist the athlete.
- A distributed approach, where different elements carry out different tasks. A WSN performs environment monitoring, whereas the user equipment monitors the athlete. Data processing is performed either by some external element or by a node of the WSN.

It can be argued that commercial smart-phones may be used in a user-centric implementation, due to the their high computational capabilities (in fact, most methods evaluated in this thesis for the decision engine requires a very low computational power, see Chapter 3). The major drawback of smart-phones is their scarce sensing possibilities. Usually, they do not have sensors to monitor environment (temperature, humidity, etc.). Moreover, some environmental data are difficult to collect accurately by a mobile terminal (*e.g.* wind speed).

However, some specific user terminal can be developed (as the one used in this project). In this case, particular sensor devices can be included to monitor the environmental data and the athlete. Though this approach is possible, only the data from the current position

of the athlete is available, therefore limiting the effectiveness of the assistant. A solution could consist of using the last available data for a position, if the environmental conditions change slowly (*e.g.* temperature). But there will be uncertainty in the places not visited by the athlete. If more than one athlete is training it is also possible to share environmental data among users, transmitting these data to a server via a cellular or wireless link connection.

The distributed approach combines static sensors with mobile ones, thus, enriching the data available to make decisions. Nevertheless, the cost of deploying a WSN must be considered. The overall costs of the equipments for the prototype deployed in the case study (see Section 2.6) was around 5000 €. In a permanent installation, for example in a stadium, this cost is negligible in comparison with the cost of the buildings. For a temporary one, a clear requirement in the design is that the nodes auto-configure themselves for any arbitrary topology, to minimize installation costs. As an estimation for the user-centric approach, the cost of the user terminal developed in this project is around 1000 €.

In this thesis the distributed approach has been selected. The reason is twofold. First, the goal of this prototype is focused more on developing and evaluating a suitable decision engine, rather than on developing advanced user equipments. Second, the quality of the data from the environment may affect the operation of the decision engine. In this case the WSN provides accurate data to test our proposal.

Next section describes the architecture and protocols of the system.

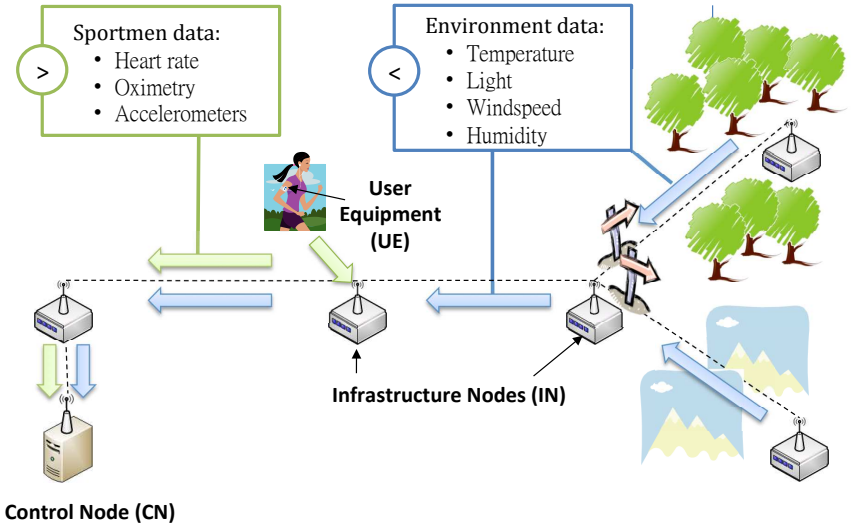
2.4 Network components

Figure 2.2(a) shows a typical system deployment. It consists of the following elements:

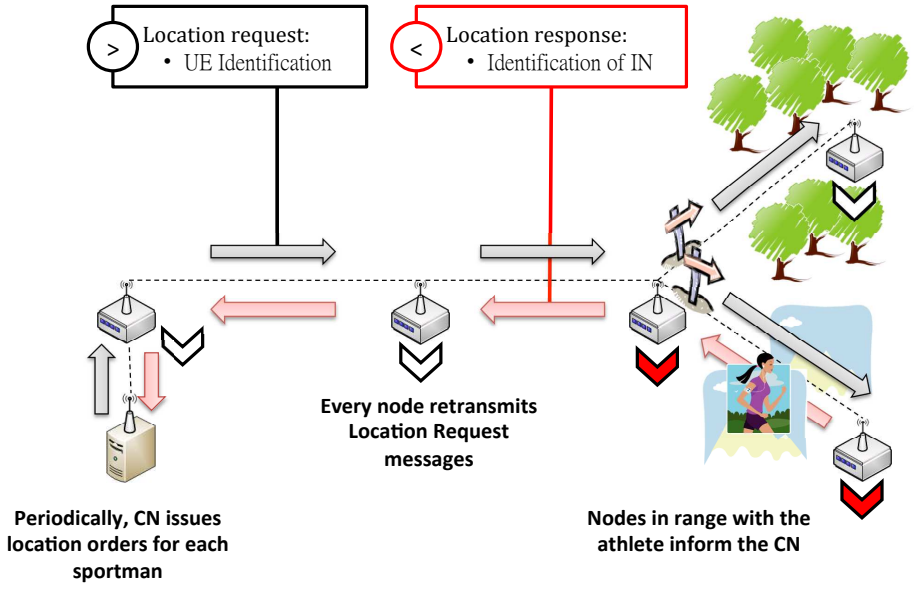
- *Infrastructure node* (IN), which cover the training area. These nodes sense environmental variables and relay data to/from the User Equipment or other INs to the Control Node. Communication relies on a wireless link. In addition, IN nodes also serve as an auxiliary network to perform athletes location in the training area. An IN is composed by a processing unit, a wireless interface including directional communication antennas, and a power supply. Directional antennas increase the communication range between IN nodes, so that less nodes are necessary, thus reducing network installation costs.

- *User equipment* (UE). This is the device carried by the user, which includes a wireless transceiver to communicate with the IN, and sensing elements to monitor athlete physical parameters. The UE also includes one or more user interfaces (acoustic, visual, etc.) to deliver training orders. The UE has the same functional blocks as an IN, but, as its weight and volume are constrained, internal antennas must be used, with a low range since they are less directional. However, precise aiming is not required for low directional antennas, and thus the UE is less obtrusive for the physical activity.

- *Control node* (CN). It can be considered as a particular IN, which collects data from all IN nodes and the UE, and analyzes them to assist user training. CN processing



(a) Environmental and athlete data acquisition



(b) Location procedure

Figure 2.2: Data flow of the sensed data, the localization information.

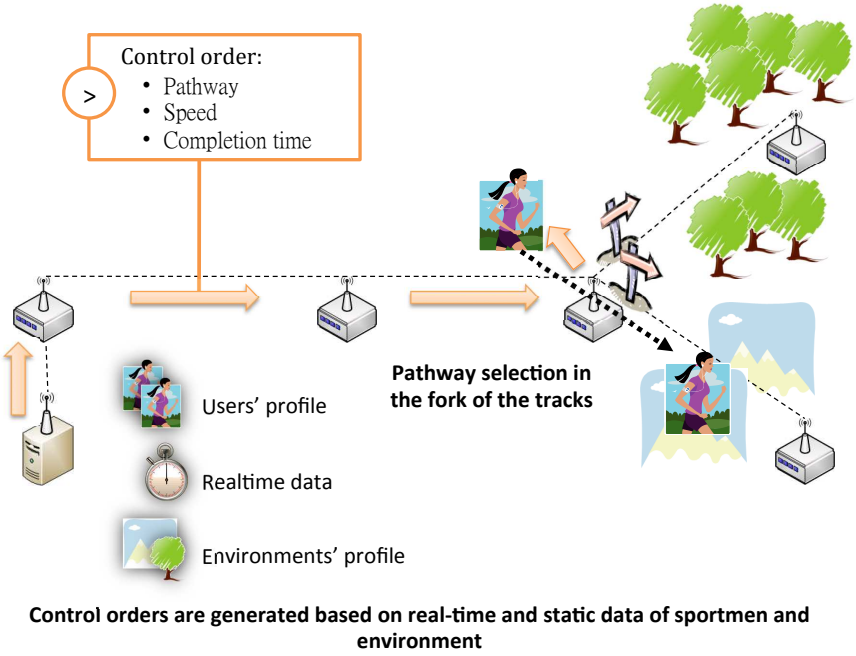


Figure 2.3: Data flow of the commands from the CN. Decision transmission.

requires access to the static information set, which is stored in databases or standard files.

2.5 Network topology and communication protocols

The scenario described in the previous section is a WSN with fixed sensing elements (IN) and mobile ones (UE). As in a typical WSN, energy constraints are demanding for these elements, because they usually run unattended. Thus, power saving is a mandatory requisite of protocol design.

In order to reduce the burden of protocol signaling, it is assumed that the IN nodes are arranged in a tree topology (data flows from the CN to the INs, and *vice versa*, suggesting also a topology of this type with the CN as the root). This topology simplifies routing task and can be easily computed at network startup (and updated periodically). Several protocols have been proposed for this task (*e.g. level discovery protocol* (LDP) [GFJ+09]). We implemented a simple version of this protocol that updates topology every hour, using the CN as the coordinator and the root node of the tree.

Notice that communication ranges differ between IN-IN and IN-UE links, since UE antennas are less directional than IN ones, creating *shadow areas*, where UE can not communicate with the WSN. This effect has been taken into account in the design of data communication and location protocols, which are described next.

The MAC protocol used is a variation of B-MAC [PHC04]. B-MAC provides unicast

and broadcast transmissions. B-MAC is focused on energy consumption minimization, which is mandatory in a real WSN deployment. Unicast transmissions include an *automatic repeat request* (ARQ) mechanism that guarantees packet delivery in case of wrong packet transmissions. Notice that transmissions between INs and UEs in shadow areas are automatically corrected by ARQ.

The position of the INs must be selected to guarantee network connectivity. For network planning purposes, there exist some specialized tools, but they usually ignore practice-related considerations (for instance, if wind sensing is interesting, some INs in open ground positions are desirable). An optimization method to determine the best sites, while guaranteeing connectivity is described in [VACRBD+08]. Moreover, the procedure described in Chapter 4 can be also used for this task.

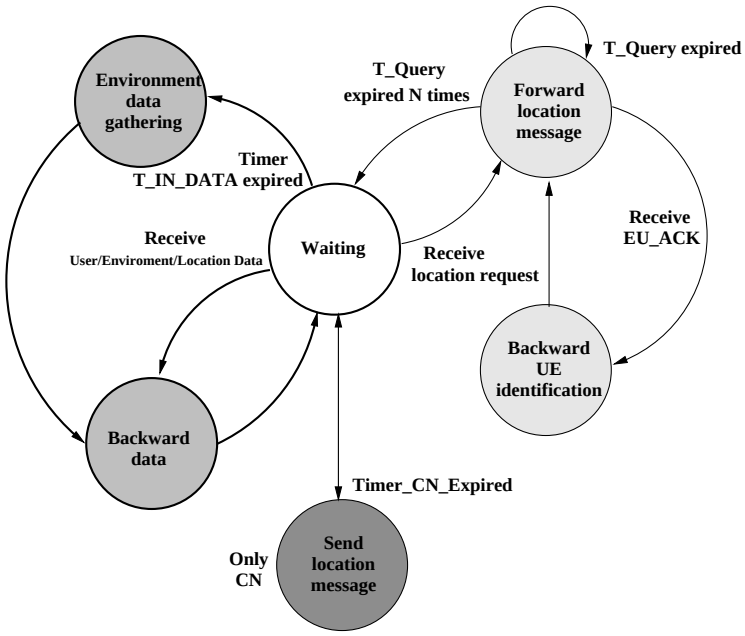
2.5.1 Data exchange

Figures 2.4(a) and 2.4(b) show the state diagram of the UE and the IN, respectively. Data is captured by the INs and UEs periodically (every T_{IN_Data} and T_{UE_Data} seconds respectively). Once the data are collected, the INs send them to their parent nodes (see Figure 2.2(a)), which backward them again until they eventually reach the CN. To reduce communication cost, the UE processes data internally to compute their moving averages and standard deviations. The resulting packets with those statistics are delivered to the CN via the INs when the UE responds to a location update query.

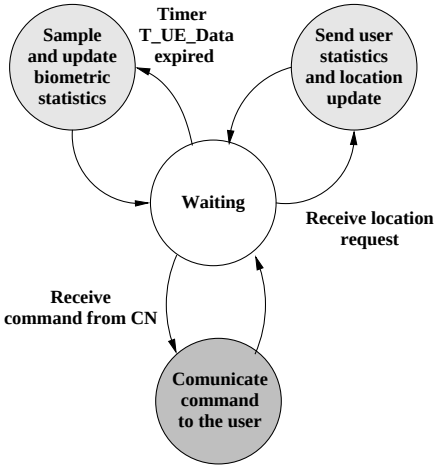
The CN issues orders for specific UEs by means of command packets (Figure 2.3(a)). These commands are delivered when the athlete must undertake some action, such as selecting another track. The CN delivers the order by transmitting the command to a specific IN, which, in turn, handles its transmission to the UE. The specific IN is selected according to the UE position and its movement direction. Both informations are provided by the location procedure.

2.5.2 Location procedure

The network also determines athletes location. Figure 2.2(b) illustrates this process. It operates as follows: the CN periodically starts a search sending a location request message, including the identity of the nodes it looks for, or a special value indicating that all nodes must respond. Immediately, the INs forward this query to all their leaf nodes using broadcast packets. If one or more of the sought nodes receive this packet, they answer to the IN with a unicast packet, which also includes the last athlete statistics. Then, the IN backwards the packet to the CN. Each IN retransmits the broadcast query packet every T_{QUERY} seconds for N_{QUERY} times, since the broadcast packet can be lost if the athlete lies temporary in a shadow area (broadcast messages are not protected by ARQ). Finally, the CN receives the information of all the INs which detected the searched node. The athlete movement direction may also be computed by the CN from the previous location records.



(a) State machine of Infrastructure Node

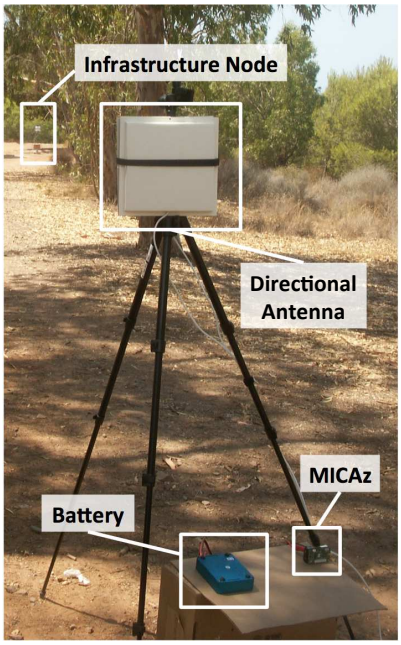


(b) State machine of User Equipment

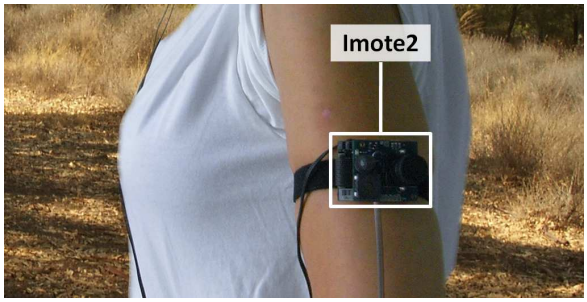
Figure 2.4: State machine diagrams.

2.6 Implementation

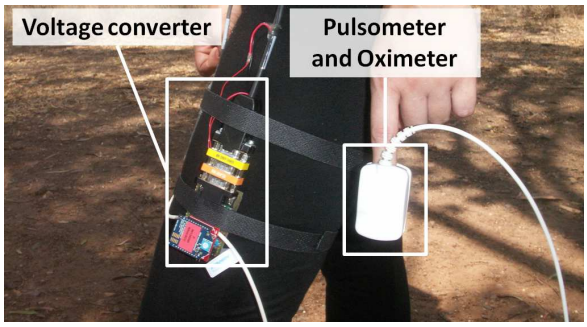
A prototype based on the architecture described in the previous section has been implemented using standard WSN hardware (MICAz and Imote2 motes from Crossbow Technology [Croa]). This prototype has been tested in a cross-country running application (described in Section 2.7). Nevertheless, it can be easily adapted to other sport activities, such as cycling, walking, etc. In the following paragraphs the implementation of each component is described, as well as the parameter selection for the protocol.



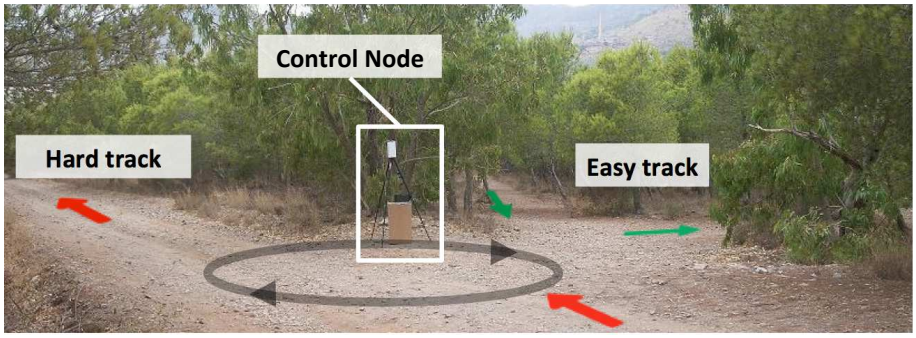
(a) Infrastructure Node



(b) Upper User Equipment



(c) Lower User Equipment



(d) Crossroad

Figure 2.5: Deployed hardware.

2.6.1 IN implementation

Each IN requires sensing, processing, and communicating capabilities, as stated in Section 2.3. We selected the MICAz mote [Crod], based on the CC2420 chip [Tex], as the IN core. It works in the 2.4 GHz band, and it is compliant with the low power Zigbee/IEEE 802.15.4 [zig] physical interface. MICAz software relies on the open source event-oriented TinyOS operating system [GFJ+09], which is a reliable platform for ad-hoc protocol programming.

Environmental sensing is carried out by a MTS400 sensor board [Croe], designed for Crossbow motes, which measures light, temperature, humidity, and barometric pressure, although in our testbeds (Sections 3.2.2, 3.2.3.1 and 3.3.2 we only activated temperature sensing.

In addition, each MICAz is equipped with two panel antennas (a 2.4 GHz Stella Doradus 24-8080 planar antenna [Ste]) to increase communication range. The reason is

twofold: (i) to cope with radio propagation issues (caused by natural obstacles like knolls, woodlands, etc.) which may worsen communication among INs; and (ii) to increase communication range between the stations, allowing a lower density of INs, and therefore, reducing installation and operation costs. In actual deployments, it was verified that communication range may reach up to 180 meters, although the INs were installed with a maximal distance of 150 meters in between, as a security margin and to increase the precision of the location algorithm. Notice that, with the default antennas that are embedded in the MICAz motes, the typical range is less than 100 meters.

Figure 2.5(a) shows a picture of two IN in a running track 2.5(d).

2.6.2 CN implementation

The CN can be considered a special IN, because it has the same sensing and data transmission capabilities. Moreover, the CN also implements the *intelligence* of the system: it issues commands to the network and to the users in order to fulfill training goals. Its decisions are transparent to the user, which only receives the training advice through his/her UE. For instance, the decision engine may select the best path to achieve a target HR, or the best to accomplish the closest with the whole session. Chapter 3 describe the goal and the procedure to compute de decision.

The location of the CN in our network deployment is shown in Figures 2.5(a) and 2.5(d).

2.6.3 User equipment

As in the IN case, the UE combines sensing and communications functionalities. Nevertheless, the UE is designed to sense human biometrics, instead of environmental parameters. Human biometrics variables usually change faster than environmental ones. Therefore, sample frequencies must be higher. In addition, to avoid continuous packet transmission, these samples are processed in the UE, and only their statistics are transmitted. The UE also delivers training advices to the athlete. Thus, the UE is the interface between the system and the user. In the implementation, audio feedback was used. However, audio messages for the user are not actually transmitted over the WSN. Instead, the UE contains the different possible commands already stored in its flash memory (several hundreds of voice commands can be stored in this 32 MB memory). The CN only sends the identification of the message, and the UE reproduces this message. Audio feedback allows users to receive complex commands, not just binary orders. Since only the message identification is transmitted, the required bandwidth and energy to send the command is minimal.

To perform these operations, the UE is composed by several modules, as shown in Figures 2.5(b) and 2.5(c). The main module is the Crossbow Imote2 IPR2400 [Croc], a wireless sensor network platform that can be expanded with extension boards to customize the system to a specific application. Imote2 also includes an 802.15.4 radio (CC2420) with a built-in 2.4 GHz antenna for IN-UE data communications. A IMB400 multimedia board [Crob] is used with the Imote2, it includes an audio output to play the speech messages, *e.g.*, "*select track* #2"; there is one message recorded per command in the Imote2 memory.

Parameter	Value
T_{QUERY}	13 s
T_{CN}	60 s
$T_{IN_{DATA}}$	120 s
$T_{UE_{DATA}}$	1 s
N_{QUERY}	3

Table 2.1: Protocol parameter selection.

The IMB400 also has a CCD color camera and audio capture, which could be used in future system applications.

For human biometrics sensing, an integrated pulse oximetry device (iPOD model 3211, from Nonin Medical company [Non11]) was used. This sensing device takes measures of heart rhythm and oxygen level with a low power consumption. The iPOD was chosen due to its lightness, size, and easy integration in the UE (via a RS-232 interface). A specific driver in TinyOS for connectivity and iPOD control was also developed. One advantage of this device is that it automatically discards wrong samples of the HR, avoiding HR sample errors (actually, iPOD returns an error code if the HR was not sensed). In our experiments the rate of successful returned samples was usually higher than 90%. In some experiments (*e.g.*, if the user temporarily moves the iPOD) it drops, but since HR is measured with a high sample frequency, enough data is always collected.

2.6.4 Protocol parameters selection

In order to tune protocol operation, different parameters and timers (introduced in Section 2.3) must be established. Some of them are related to the athlete activity (*e.g.* speed), whereas others are related to environmental change rate (the sampling periods) in order to limit message exchange to minimize the impact on energy consumption. The selection of parameters is discussed next. Table 2.1 summarizes the parameters for the specific examples of Sections 3.2.2, 3.2.3.1 and 3.3.2.

- T_{QUERY} is the elapsed time between IN transmissions of location update requests. It is related to the shadow time, T_{SHADOW}, which an athlete may experience between consecutive INs. The criterion was $T_{QUERY} = T_{SHADOW}$, since this guarantees that, in the worst case, the number of lost packets is just one. Otherwise, for lower values, more packets may be transmitted (and lost) while the user is in the shadow area. For higher values, the delay of the communication between the IN and the UE will increase.

  To compute T_{SHADOW} let us denote v as the average expected user speed, and D_{SHADOW} as the length of the shadow area. Then,

$$T_{QUERY} = T_{SHADOW} = \frac{D_{SHADOW}}{v} \tag{2.1}$$

Besides, D_{SHADOW} can be computed as a function of IN antenna directivity gain (G_{IN}), the directivity gain of the UE (G_{UE}), IN output power (P_{IN}), UE output power (P_{UE}), and the distance between INs (D).

$$D_{SHADOW} = D(1 - 2\sqrt{\frac{G_{UE}\, P_{UE}}{G_{IN}\, P_{IN}}}) \tag{2.2}$$

Hence,

$$T_{QUERY} = \frac{D(1 - 2\sqrt{\frac{G_{UE}\, P_{UE}}{G_{IN}\, P_{IN}}})}{v} \tag{2.3}$$

For a reference training speed of 15 km/h (4,16 m/s) and since in our testbed G_{IN} = 11,15 dB, G_{UE} = 1,41 dB, P_{IN} = 0 dBm, P_{UE} = 0 dBm, and D = 150 meters, then $T_{QUERY} \approx 13$ s.

- N_{QUERY} is the number of times that an IN tries to locate UEs nearby. It is configured to be the expected number of intervals of T_{QUERY} s that a user stays in the IN range:

$$N_{QUERY} = \frac{D}{v}\frac{1}{T_{QUERY}} \tag{2.4}$$

With the reference configuration, we select N_{QUERY} = 3 (the actual value 2,77, but the parameter is an integer, so it is rounded up).

- T_{CN} is the elapsed time between two location procedures initiated by the CN. The minimal interval corresponds to the expected time in IN range:

$$T_{CN} = N_{QUERY} T_{QUERY} = \frac{D}{v} \tag{2.5}$$

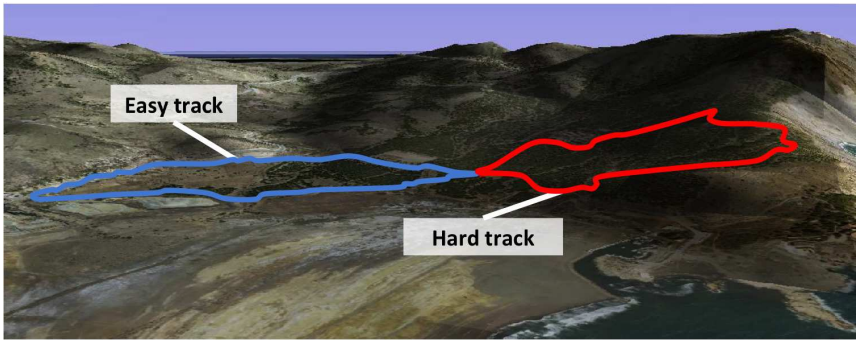
Higher values reduce energy consumption, at the expense of a lower location accuracy. With the reference configuration, the minimal value should be $T_{CN} \approx 36$ s.

2.7 Prototype deployment

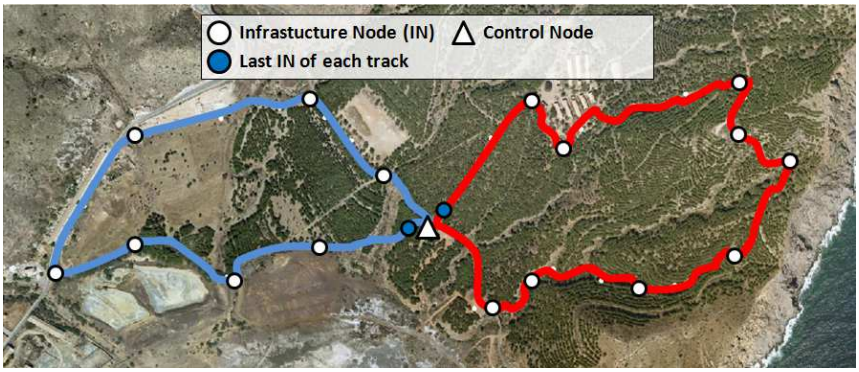
The prototype has been deployed in a open-area ready to perform cross-country training experiments. This section focus on the network deployment configuration; the decision part and the complete validation experiments are addressed in Chapter 3 of this thesis.

The selected area is located near Cartagena (Spain), and it is shown in Figure 2.6. It consists of two interconnected loops (red and blue) with different hardness and environmental conditions due to:

- Closeness to the sea in some areas of the circuit. For example, training by the coast is characterized by constant winds.



(a) Cross training circuit



(b) Deployed infrastructure

Figure 2.6: Aerial sights of the training circuit.

- Different terrain slopes, since part of the circuit is on a hill whose height is 97 meters over sea level, with some slopes of 14%.
- Shadow, depending on training hours and the trees along the circuit (as can be observed in the red circuit in Figure 2.6(a)). It has influence on temperature.
- Different lengths: 1.1 and 0.9 kilometers for the red and blue tracks, respectively.

The *red* track is considered *hard* due to of its length and more pronounced slope. The blue track is considered as *easy*. There are ten deployed INs in the difficult sector of the training area and eight in the easy one, as shown in Figure 2.6(b). The CN is placed in the junction of both loops. The network was configured with the protocol parameters that are summarized in Table 2.1.

In the deployment, communication was extensively tested and performs well in all the tests. The ARQ algorithm in the MAC layer cope with retransmission errors (mainly produced when the UEs were in a shadow area), and automatic retransmissions of broadcast packets solves concerns with location protocol. Simultaneous test were carried out with three runners, and protocols operation were again correct. In this case the CN order a general location (all UEs must respond) in location request orders. The test were limited to three users since this is the number of Imote2 available, but we expect similar performance with a large number of users. However, additional test should be performed to confirm this hypothesis.

Besides, some parameters were measured in our test, as a reference of the protocol operation: The round trip time of a message from the CN to a particular UE (or vice versa) was less than 0.55 ms per hop for a packet length of 10 bytes including headers (the length of a command packet). The average location delay was 5.73 s. (the procedure may suffer temporary fading of the communications due to shadow areas). And, finally, in the location procedure, an average number of 1.3 nodes detected each UE (note that more than one node can receive replies from the same UE).

2.8 Summary

This chapter introduced the networking and architectonic aspects of our AmI system oriented towards sports. Many of the design decisions are common to any AmI scenario, and include topological and communication constraints. We implemented our system in real devices, and it has been deployed in a prototype for monitoring runners on a cross country circuit. Biometric data, as well as environmental measurements, take place in this experimental test-bed, which has been used to develop suitable decision-making engines. This latter part is discussed thoroughly in the next chapter.

Chapter 3

Decision making in AmI sport environments

The AmI system presented in the previous chapter represents a great technological advance in personal training systems: the user can be easily located in the training field, her/his biometric variables and the changing environmental data are available in real-time to monitor the training activity. Moreover, the AmI system allows for the delivery of personalized detailed instructions to the athletes about how to proceed with the training program. These decisions must be computed by an intelligent component (CN) of the AmI prototype. The decision-making process receives as inputs the sensed data from the environment as well as the athletes' biometry. Based on these data the goal consists in selecting the best tracks in the circuit to monitor the HR of the athlete within a given interval.

This chapter analyzes a set of decision-making methodologies that have been applied to this decision-making task. We study two broad types of decision-making procedures: (i) optimal classifiers, and (ii) dynamic programming. The former is able to decide the optimal decision for the next training stage, whereas the latter considers the optimization problem globally, leading to the selection of suitable policies for the whole training period.

3.1 Problem characterization

Decision making is an inference process that requires the use of input data to compute the necessary outcome. As it is depicted in the system approach of Figure 2.2, the input set for the problem are the athlete biometry and the environmental conditions. For the sake of simplicity, in our work we select as inputs the influence of temperature [BvSNP03] and track hardness [KV88], and the real-time HR of the user. However, the methods developed in this chapter may be easily expanded to include new input variables.

In our testbed, athletes train (we exclusively consider a running exercise) in a field with track alternatives. Each track has a different *hardness* degree and the weather conditions (*temperature*) may vary along the training. The goal is to select a suitable track for the runner at each path junction, in order to fulfill an overall training program. In the prototyped system the aim is at controlling the HR of the user. Different training HR

Loop	1	2	3	4	5	6	7	8	9	10	11	12	13
Hardness	WU	E	E	E	H	E	E	H	H	E	H	H	H

Table 3.1: Course profile of a training session.

ranges are possible (*e.g.* aerobic, weight control, etc.). Before training starts, athletes or coaches select the desired HR profiles during exercise, *e.g.* "perform all tracks in fat-burn HR range". Henceforth, let us denote this kind of training as *homogeneous*, as it involves a single HR ranges. On the other hand, if training involves different HR ranges, we will call it *heterogeneous* (*e.g.* "perform half of the tracks in fat-burn regime and the other half in cardio-training").

As training conditions can rapidly vary, the system would be expected to make real-time decisions to meet the athletes needs. Therefore, constant user and environmental monitoring are a requirement. The WSN described in Chapter 2 is in charge of providing these data in real-time.

This prototype has been validated in a real testbed deployment (see Section 2.7). Let us recall that it consists of two interconnected loops ("easy" and "hard") with different hardness and environmental conditions due to the slope, length and the shadow at different times of the day.

The knowledge base required by the decision engines was collected by repeating trials of a training session in the previous circuit. The data recorded were the HR (measured in beats per minute), together with the temperature ("high" if average temperature in the track is over 25 Celsius degrees or "low" for lower ones). Each session comprised the following loops/stages:

1. There was an initial *warm-up* period (WU). During it the athlete trains in the easy sector of the circuit. The data from this sector is discarded.

2. Each training session consists of 12 loops in the circuit (after the warm up sector). At each loop the athlete ran either in the hard or in the easy track. The order was the same for all the sessions (Table 3.1), and it determined the *course profile*. Besides, the ambient temperature was also monitored. The sessions were performed at the same hour in different days.

Usually HR is divided in different intensity levels or classes according to the type of training activity [HL+07, TIH+07]: The HR in each class is in a percentage range of the HR_{max} recommended.

- VO2 MAX (Maximum effort): [90%,100%].
- Anaerobic (Hardcore training): [80%,90%].
- Aerobic (Cardio training/Endurance): [70%,80%].
- Weight Control (Fitness/Fat burn): [60%,70%].
- Moderate activity (Maintenance/Warm up): [50%,60%].

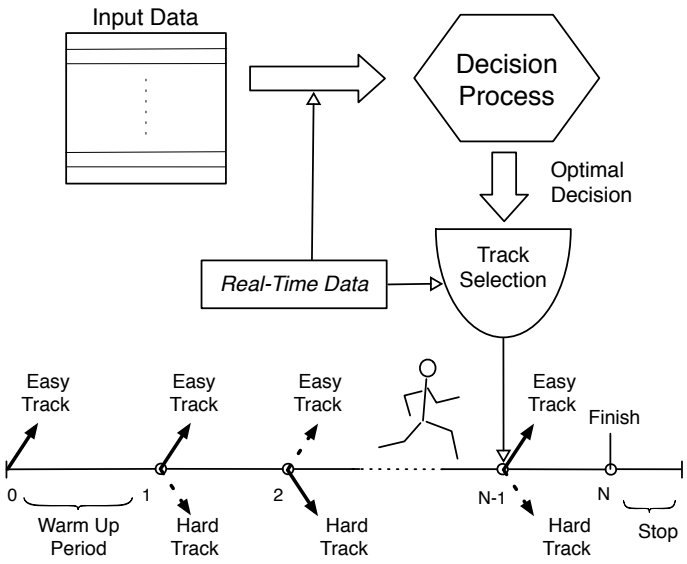


Figure 3.1: Decision process scheme. At each decision instant the classifier compute for every possible track selection its training profit, and then selects those with the maximal one.

An accurate formula to compute the maximal HR [TMS01] for a healthy male person it follows the relationship:

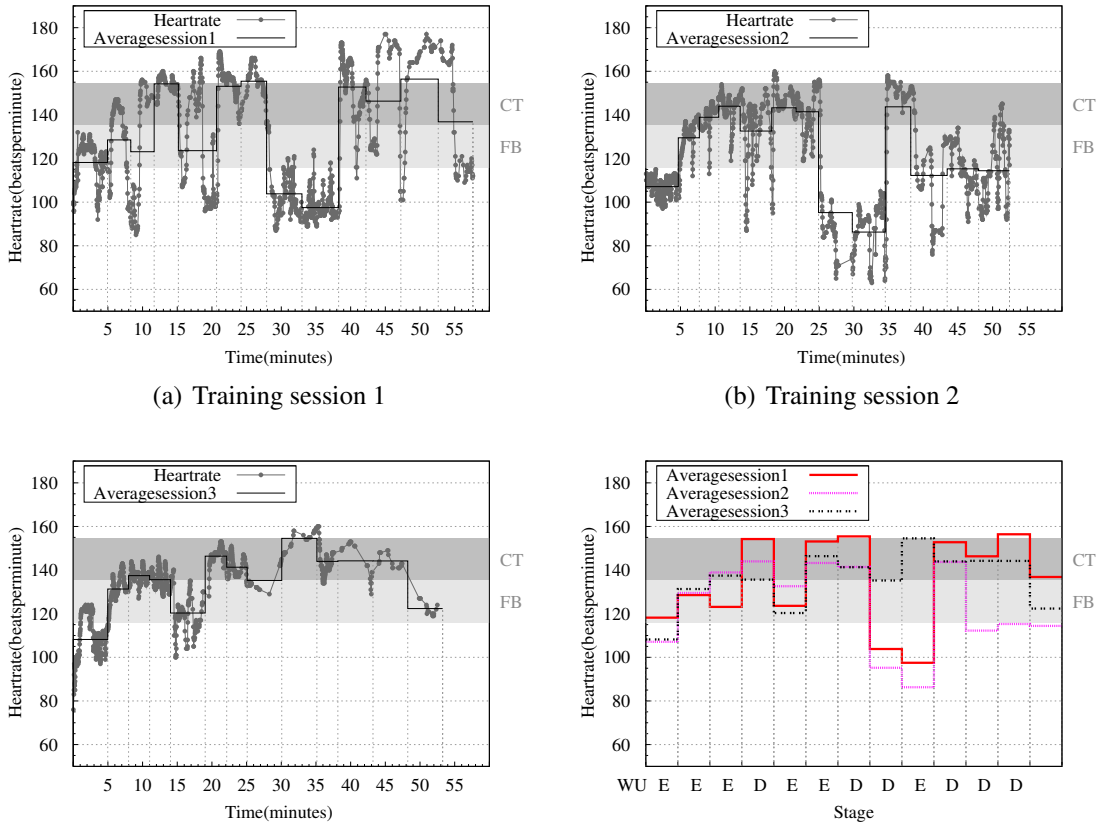
$$\mathrm{HR}_{max} = 208 - 0.7 \times age \tag{3.1}$$

The HR will depend on several aspects such as athlete's condition, elapsed training time the hardness of the track, environmental conditions, and so forth. From these variables (*features* in decision-making terminology), the system must select a suitable output (next track). Figure 3.1 depicts the general decision-making procedure.

However, the HR of a runner and, in turn, his/her performance, is difficult to characterize. Next we intend to show the difficulty of dealing with such a complex bio-parameter as human HR can be and present some previous investigation efforts.

3.1.1 HR evaluation

The problem of evaluating the expected HR is complex. Analysis of HR dynamics by methods based on chaos theory and nonlinear system theory have received attention recently, due to observations that suggest that the mechanisms involved in cardiovascular regulation likely interact with each other nonlinearly [KFP+91, ASBG99, HMP03, TCRB12]. Thus, characterizing HR and athlete performance is even more difficult. As an example of its difficulty, Figure 3.2 shows results for three training sessions of an athlete, which were performed as part of the experimental validation of methods presented in Sections 3.2.2, 3.2.3.1 and 3.3.2. Most of HR samples taken belong to cardio-training (CT) and fat-burn (FB) regimes. Figure 3.2(d) shows average HR evolution with training stage; after a warm-up (WU) period follows several stages with two kinds of exercise endurance: hard (D) and easy (E). It is important to observe in Figure 3.2(d) how the same



(a) Training session 1

(b) Training session 2

(c) Training session 3

(d) Training sessions average

Figure 3.2: HR evolution in three training sessions.

part in different training sessions produces so different average HR values. This is because athletes psychology and physical condition have an important role in performance. They can induce the athlete to increase physical activity, and, accordingly, HR in the easy section of the course; or the athlete to rest in the same section, leading to activity and HR decrease. For example, this is possibly the reason why session 3 ends several minutes before sessions 1 and 2 do. Note as well that HR increases are not directly related to hardness or temperature increases: *e.g.* the hardness increase from stage 7 to stage 8 (see Figure 3.2(d)) produces a sudden HR drop.

There exist several numerical and statistical methods to determine the relationships hidden in complex processes where many variables can be involved. These methods have been utilized to predict physiological parameters, such as HR, systolic blood pressure, or body temperature. Some possibilities are: *support vector machines* (SVMs) which have been successfully applied to medical decision support. Reference [VCC99] describes how to mine medical knowledge from time series of high-dimensional numerical data describing patients in intensive care. A SVM is used to learn how and when a drug dose must change. Also, in [BJM04] it is proposed to improve the diagnosis of tuberculosis infections by means of SVM image classification. Nevertheless, for sportsmen training purposes, a SVM statistical method requires a huge amount of time series data of many training sessions, in order to feed the knowledge base of each athlete appropriately. There-

fore, this approach becomes impractical. Based on HR and 3D acceleration signals the strategy in [YJ08] employs *feed-forward neural networks* (FFNN) to predict the next time step in the HR sequence. The predicted HR follows the variance of the real HR, although with noticeable differences in some cases. In [LJTL08] a dynamic heart rate prediction model is also used by a predictive controller to optimize the cycling training.

Recently, the statistical tool k-Nearest-Neighbors (k-NN) is has been successfully used as a classifier to predict a variety of heart diseases. For example, since there is a direct relation between fluctuations of oxygen saturation in blood and variations in HR, in [QMAHTG+09], the authors use the k-NN classification method to detect obstructive sleep apnea. Also, in paper [AAA09] a pruned variation of a k-NN classifier is proposed to recognize different types of arrhythmia beats. Numerical approximation techniques based on splines are extensively applied in signal processing and surface fitting areas [LS86, Uns99, LdSPPT01]. None of those techniques, k-NN or splines, has been applied to ambient intelligence systems, and either splines to any classification problem. Section 3.2 illustrates how those mathematical tools can be applied to sport training AmI system.

3.1.2 Outline of this chapter

In the remainder of this chapter we analyze several decision-making methodologies utilized to implement the intelligence (*i.e.* the track selection procedure) of our AmI training system for outdoor sports. Three classification techniques have been investigated:

- k-Nearest-Neighbors (k-NN) [Bis06].
- Splines of (m, s) order [LdSA89].
- Dynamic programming (DP)

Whereas the two first classification techniques make decisions for a *single-step* horizon, *i.e.* they look for the best track decision to fulfill *only* the next step in the training program, the DP methodology allows to select a series of tracks which maximizes the correct distribution of the HR intensity levels during the whole training period. This is a *multi-step* decision-making process, since it is assumed that decision at step n will affect HR at steps $n + 1$, $n + 2$, and so on. The latter classification method allows to design heterogeneous training sessions (involving more than one HR range), rather than homogeneous ones allowed by single-step classifiers.

Next two sections, 3.2 and 3.3, describe the key ideas of single-step and multi-step decision engines, respectively.

3.2 Single-step decision engines

3.2.1 Features and objective

The following features are considered in our decision engine implementation:

- F1: Current track environmental temperature, which is considered to be divided into two ranges: "high" (1) if average temperature in the track is over 25 Celsius degrees or "low" (0) for lower ones.
- F2: Next track temperature: High (1) or low (0).
- F3: Current track hardness: Hard (1) or Easy (0) tracks.
- F4: Next track hardness: Hard (1) or Easy (0).
- F5: Average heart rate during the current track (in beats per minute, bpm).
- F6: Variance of the heart rate during the current track (in bpm^2).

The goal of the single-step classification engines consist in maintaining the runners' HR in a given target range. For our experiments, aerobic (cardio-training) was the selected target.

3.2.2 Solving the decision-making process with (m, s)-splines

In this case, the decision engine implementation relies on (m, s)-splines technique introduced in Section 1.3.2.1. Figure 3.3 depicts the decision process as described next. For each user, a database contains a knowledge base (represented as the input table to the "Spline function computation" box in Figure 3.3) with recorded sets of these variables and the associated HR value) measured in the experiments. These values are used to construct an interpolation function, which is evaluated for the current conditions, and returns the expected HR. It should be noted that the decision engine behaves as a learning machine. This is possible because new sensed data (HR) is available at the end of a track (*e.g.* at t_{i+1}). Then, captured values are incorporated in the data-set, and can be used to re-compute a new spline function. These feedback is represented with a dashed line in Figure 3.3. The feedback improvement is evaluated in the Section 3.2.2.2.

For every possible next track, based on the input parameters (hardness, temperature, etc.) the CN estimates the expected HR ("Spline evaluation box" in Figure 3.3) using the technique of (m, s)-splines. Then, the CN selects the appropriate tracks with the following rule:

- If the expected HR is in the target interval, the track is suitable.
- Otherwise, the track is classified as not suitable.

If there are several suitable tracks, the choice is random. If none is valid, the system select the track with minimal difference between the expected HR and the mean value of the target set.

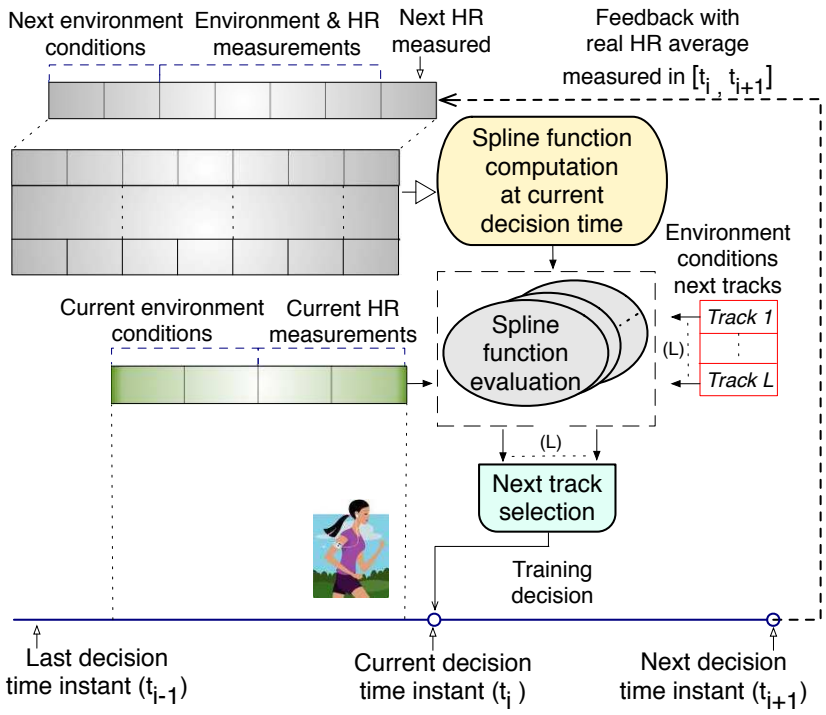


Figure 3.3: Decision engine operation based on (m, s)-splines.

Next		Current				Measured
Hardness (x_1)	Temperature (x_2)	Hardness (x_3)	Temperature (x_4)	Average HR (x_5)	Variance (x_6)	Average HR
0	0	0	0	124.06	200.57	134.78
0	0	0	0	131.52	738.09	139.61
1	0	1	0	108.78	323.5	110.69
0	0	0	0	131.29	200.93	125.67
0	0	1	0	127.59	635.41	148.76
0	0	0	0	148.76	183.11	151.86
0	0	0	0	128.55	312.67	123.14
0	0	0	0	153.13	165.6	155.44
⋮						⋮
0	1	1	1	81.19	455.38	150.49
0	1	0	1	150.49	297.94	150.55
0	1	1	1	90.66	83.74	142.76
0	1	0	1	137.45	11.89	135.62
0	1	1	1	120.3	96.18	146.39
0	1	1	1	154.56	12.41	143.99

Table 3.2: Knowledge base data-set example.

3.2.2.1 Example of operation

As soon as the CN detects a UE in the IN before a track fork (decisions instants), it executes the method described above. In this section we provide a detailed example of the operation. The knowledge base is represented in the next table (which contains real values obtained during validation of the case study):

Let us assume that a new track must be selected, that user is 35 years old and its intended activity is “cardio training”. Then, the target HR is in the interval (129.5, 148)

Next		Current			
Hardness (x_1)	Temperature (x_2)	Hardness (x_3)	Temperature (x_4)	Average HR (x_5)	Variance (x_6)
0	0	0	0	140.08	24.36
1	0	0	0	140.08	24.36

Table 3.3: Input data points for the next training choice.

bpm. In addition, let us suppose that input data corresponds to values given in Table 3.3. That is, there are two possible options, one difficult and one easy track, both with cold temperature. Obviously, the current HR parameters of the athlete are the same. Then, the interpolation function is computed. In this example, we set $m = 3$ and $s = 1/2$. The interpolant of this example is:

$$\begin{aligned} 148.36 - 82.38x_1 - 82.383x_1^2 + << 73 >> + 0.085(|0. + x_1|^2 + |x_2 - 1.|^2 \\ + |x_3 - 1.|^2 + |x_4 - 1.|^2 + |x_5 - 154.556|^2 + |x_6 - 12.414|^2)^{1/2} \\ -0.0135(|x_1 + 0.|^2 + |x_2 - 1.|^2 + |x_3 + 0.|^2 + |x_4 - 1.|^2 \\ + |x_5 - 137.451|^2 + |x_6 - 11.887|)^{1/2} \end{aligned}$$

where the $<< 73 >>$ represents a short form expression of 73 remainder terms.

Next, the two input data sets are used to evaluate the computed interpolation function. The results for the expected HR are HR=137.3 and HR=121.9, respectively. The first track is classified as "Suitable", and the second one as "Unsuitable". The first track is selected and the CN sends a command to the UE with the instruction of select this track.

3.2.2.2 Evaluating the decisions

The conformity of the classification method was measured by means of its degree of discrepancy between the decision engine outcome and the following real HR. That is, if after running in the selected track the average HR was not in the target interval, the classification failed. Otherwise, it was correct. The ratio of successful test is measured in our conformity tests. To perform these tests several samples (test points) were taken out from the knowledge base: for the first runner it included a total of 119 HR values with their associated input variable values, and 14 records were randomly chosen. For the second athlete, with 110 records, 12 were randomly chosen. These records represent 11.8% and 10.9% of their respective knowledge bases. Then, the classification method was applied and the success rate was measured. Table 3.4 summarizes the results of four conformity tests for the two athletes. The results average the ratios of 30 experiments selecting different test points.

Feedback and computational load. Tests performed with the decision engine show how an increase in the number of records increases the quality of the decision. The algorithm was tested selecting a random subset of 25% of the whole data set, than adding a second random subset up to 50% of the whole data set, another one up to 75%, and finally with the whole data set. Results for four random test are summarized in Table 3.5 for athlete-1 (for athlete-2 results are similar). Clearly, the percentage of valid decisions raise with

	Number of records	Test points	Ratio of valid decisions Test 1	Test 2	Test 3	Test 4
athlete-1	119	11.80%	76.40%	76.30%	85.40%	72.40%
athlete-2	110	10.90%	75.30%	75.00%	75.60%	83.70%

Table 3.4: Conformity test results.

	Test 1				Test 2			
Number of records	28	55	83	110	28	55	83	110
Ratio of test points	10.7%	11.0%	10.8%	10.9%	14.3%	10.9%	12.1%	10.9%
Ratio of valid decisions	33.3%	50.0%	55.5%	83.3%	50,0%	83.3%	90.0%	91.7%
(m,s)	(2,3/2)			(5,3/2)	(2,3/2)			(3,1/2)
Memory used in bytes	322584	472384	676240	3142544	317638	472072	667968	1056224
Computing time in seconds	0.178	0.225	0.318	0.864	0.175	0.232	0.309	0.442
	Test 3				Test 4			
Number of records	28	55	83	110	28	55	83	110
Ratio of test points	10.7%	11.0%	10.8%	10.9%	14.3%	10.9%	12.1%	10.9%
Ratio of valid decisions	33.3%	50.0%	66.7%	83.3%	66.8%	83.3%	88.9%	100.0%
(m,s)	(2,3/2)				(2,3/2)			(3,1/2)
Memory used in bytes	354208	525544	771688	1082816	322584	472088	676552	1056456
Computing time in seconds	0.205	0.307	0.456	0.661	0.180	0.202	0.301	0.419

Table 3.5: Conformity test and algorithm performance results for athlete-1.

the size of the data-set. Therefore, as decision engine has more records, accuracy in the prediction increases.

In addition, the computing time and memory requirements for the spline computation was measured (in a laptop computer with a 2 GHz Intel Core Duo CPU, and, 1 GB of DDR2 SDRAM). Both parameters show a linearly dependent of the data set size. The values obtained ($\leq$ 0.5s in all tests, and, near 1 MB for every 100 records) allow to compute interpolation function in real time, even, using a low end performance computer or a smart-phone. Besides, evaluation time is less than 1 ms in all cases evaluated.

Environmental parameters influence. Finally, conformity tests were carried out without environmental parameters to measure their influence in the decision. Two experiments have been performed. In the first one, the information of the temperature (current and future one) was discarded in the computation of the spline, as well as in its evaluations. In the second one, both temperature and hardness (current and future one) was removed. Results are summarized in Table 3.6, and show a clear reduction in the quality of the classification method. In the case of athlete-1 result is disastrous, with success ratios below 50%.

3.2.3 k-NN Classification

The k-NN method belongs to the category of Prototype Methods. It operates on the intuitive idea that close objects are more likely to be in the same category. Thus, k-NN predictions are based on a set of prototype examples that are used to predict new (or *unseen*) data based on the majority vote (for classification tasks) and averaging (for regression)

athlete-1						
	Number of records	Test points	Ratio of valid decisions Test 1	Test 2	Test 3	Test 4
With environment vars.	110	10.90%	83.33%	91.67%	83.33%	100.00%
Without temperature	110	10.90%	66.67%	75.00%	66.67%	83.33%
Without environment vars.	110	10.90%	50.00%	41.67%	33.33%	33.33%
athlete-2						
	Number of records	Test points	Ratio of valid decisions Test 1	Test 2	Test 3	Test 4
With environment vars.	119	11.80%	85.71%	85.71%	76.92%	85.71%
Without temperature	119	11.80%	78.57%	78.57%	61.54%	71.43%
Without environment vars.	119	11.80%	78.57%	71.43%	69.23%	64.29%

Table 3.6: Influence of environment variables on the percentage of valid decisions.

over a set of k-nearest prototypes.

The method operates as follow. Given a data set consisting of N pairs $(p_1, g_1), \ldots, (p_N, g_N)$ where p_1 is the feature information vector and g_i is the corresponding scalar class label. Each of the N pairs is called *prototype*, or *scenario*, and, every data p_i belongs to the feature space of dimension P. In our experiment specification a feature space of dimension 6 consisting of the previous features. Besides, two classes are possible ("In range" or "Out of range") corresponding to a track where the HR is in the objective range or outside, respectively. Thus, given a query point p_0, the classification problem consists of finding the k training points $p_{(r)}, r = 1,..., k$ closest in distance to p_0, and then classify it, (*i.e.* assign a class label) using majority vote among the k neighbors. If vote results in a tie then it would be resolved at random. In our study, it is assumed that the features are real-valued, and, Euclidean distance is used in the feature space:

$$d_{(i)} = \|p_{(i)} - p_0\|_2 \tag{3.2}$$

Besides, feature values must be scaled so that the arithmetic mean for each of them is zero and its standard deviation is 1.

3.2.3.1 Adapting the method

Figure 3.4 shows operation of the decision engine. The input data is used in the *distance computing* and *majority selection* blocks. At each decision instant (t_{i-1}, t_i, ...) the classifier compounds L *query* points, one for each one of the L possible track alternatives, from current environmental variables (temperature and track hardness) and from the athlete's HR statistics (which are calculated in the *HR average and variance* box in Figure 3.4 from HR samples). Then, for each query point, the *distance computing* block calculates the distance from this query point to every point in the input data table, sorts them, counts the label classes of the k closest neighbors, and assigns the most frequent class to the query point.

Table 3.7 contains an input data sample extracted from our experiment. Class label 1 means "In range", and, label 0 "Out of range". Class "In range" corresponds with a track were at least 50% of the HR samples are in the target range. Otherwise, it is considered "Out of range".

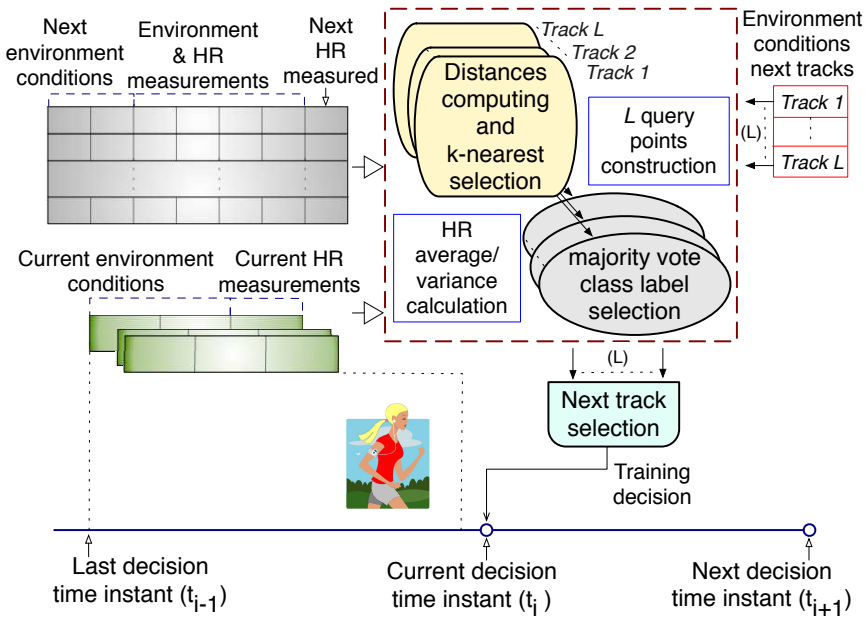


Figure 3.4: Decision engine functional scheme using k-NN classifier.

3.2.3.2 Tests results evaluation

Figure 3.5 shows the classification success ratio of the k-NN decision engine implementation for a range of $k \in [1, 30]$. These results are obtained using a data base of 86 points (50% of "In range" class versus 50% of "Out of range" class), from the same athlete, that were captured over a training period of 10 days. Query points were selected extracting at random a 15% of these samples. A point is classified correctly if the real classification (notice that this is already known) matches the outcome from the k-NN classification. Besides, k-NN uses the remainder 85% points as the input data. In Figure 3.5 the average success ratio is depicted with its corresponding confidence interval for a 90% confidence level. It can be observed that the best results appears using a value of $k = 10$, with 70% of success. This classification process was run using the same number of points for each class. Other experiments were performed using a large number of samples with unbalanced classes (20% of "In range" class points versus 80% of "Out of range" class points), but the ratio of success classifications were close to 50% (notice that a random classification achieves this level). In addition, for another athlete, with few input registers (only 7 "In range" points) the outcome of the k-NN also resembles a random classification. Therefore, we conclude that the method proposed required a balanced and large number of points of both classes.

In addition, the influence of environmental data on the success ratio was measured. Figure 3.5 shows the results if no environmental data (neither temperature nor track hardness is used) and if no track hardness is used. As can be clearly seen, both results are worse than the outcome if all the environmental data is used. We conclude that if decision is taken from only HR statistics, success percentage is 10% inferior in the considered range of k. Surprisingly, for large values of k ($k > 15$) system classification ratio increases if all environmental data is eliminated.

Next		Current				Measured
Hardness (x_1)	Temperature (x_2)	Hardness (x_3)	Temperature (x_4)	Average HR (x_5)	Variance (x_6)	Class
0	0	0	0	124.06	200.57	0
0	0	0	0	131.52	738.09	1
1	0	1	0	108.78	323.5	0
0	0	0	0	131.29	200.93	0
0	0	1	0	127.59	635.41	1
0	0	0	0	148.76	183.11	1
0	0	0	0	128.55	312.67	0
0	0	0	0	153.13	165.6	0
⋮					⋮	
0	1	1	1	81.19	455.38	1
0	1	0	1	150.49	297.94	1
0	1	1	1	90.66	83.74	1
0	1	0	1	137.45	11.89	1
0	1	1	1	120.3	96.18	1
0	1	1	1	154.56	12.41	1

Table 3.7: Data-set example.

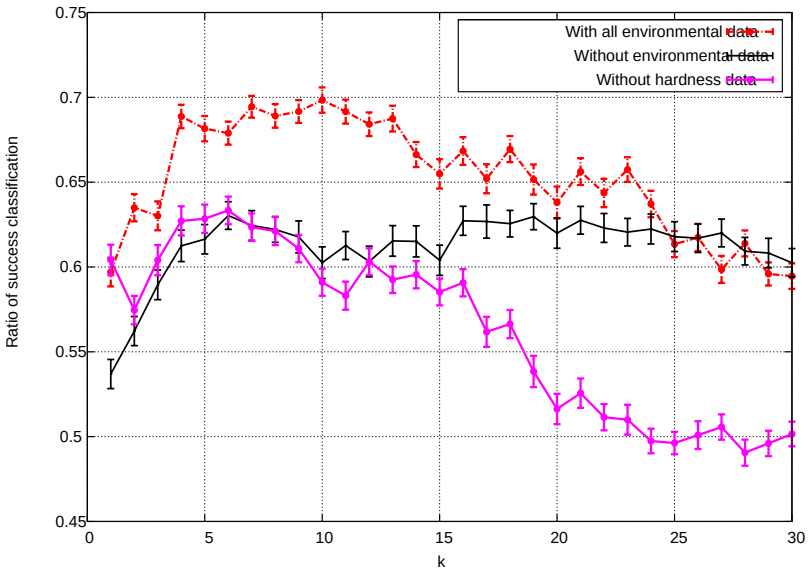


Figure 3.5: Ratio of decision engine success.

Finally, Figure 3.6 shows the ratio of classification for the different classes. That is, the probability of successful classification if the data point actually belongs to the "In range" class, or to the "Out of range" class, respectively. It can be observed that both curves decrease as k increases, with a sharper decrease in the case of incorrect classification event. Moreover, notice that two types of errors are possible in the classification:

- Error I: Do not select an "In range" track.
- Error II: Select an "Out of range" track.

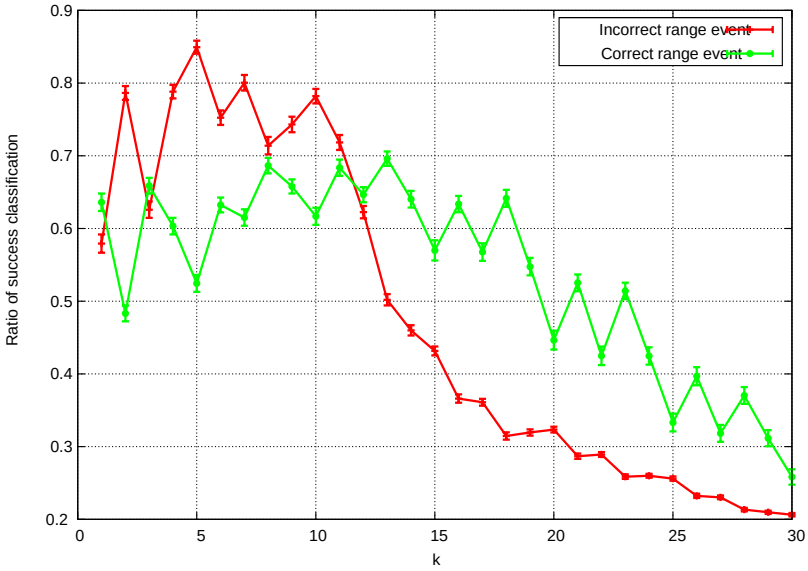


Figure 3.6: Ratio of decision engine success.

For the selected value of $k = 10$, the Error-I type is more likely than Error-II (37% of probability versus 23%), and this is beneficial for our problem: Error-II consequence is that user will be outside the selected HR range, and this situation must be avoided. However, if another track is suitable, Error-I has no consequence (since the user does not exceed the selected HR). This observation also suggests that $k = 5$ may be a good option (16% of Error-II probability versus 46% of Error-I probability). However, in this case, the number of possible track selections must be enough to guarantee that a path is selected as suitable.

3.3 Multi-step decision engine

In the previous approaches only the problem of selecting the best decision for one stage is considered, independently of the future evolution. That is, optimization is done for a single-step scenario. We consider now the multi-stage case, where the track is selected in order to fulfill training goals, considering the future evolution of the athletes. This problem can be addressed as a dynamic program. As in the previous section we assume the goal of performing all tracks within a selected HR range ($\widehat{HR}$).

3.3.1 Preliminaries and problem formulation

For our formulation, let us assume a partition of HR ranges consisting of m-non-overlapping levels, each one comprising heart rate in the set $HR^i = [hr_i\ , hr_{i+1})$ for $i = 1, \ldots, m$, where hr_i denotes the lower bound of the heart rate associated to range i.

As stated in the introduction, in our scenario, the runners train on a circuit containing several tracks (t tracks), and each track is associated to different difficulty level. At each stage, a track must be selected. Therefore, a training session consists in a sequence of

N tracks. The goal of the session may be either to perform all tracks within a selected HR range, or to fulfill a multi-goal training, for instance performing $N-4$ tracks in range HR^2 and the remaining four in HR^3. The HR will depend, among others, on the following aspects: athlete's condition, previous stages, the hardness of the track, and environmental conditions. From these variables (features in decision-making terminology), the system must decide which track to follow for the upcoming stage.

Thus, the CN must select, at each decision epoch, the best track in order to accomplish the overall training goal. Previous approaches (see Section 3.2) consider only the problem of selecting the best decision for one stage, independently of the future evolution. That is, optimization is done for a single-step scenario.

Now, we consider the multi-stage case. That is, track selection in order to fulfill training goals, considering the future evolution of the athletes. This problem can be formulated as a dynamic program (DP) following Bellman's equation [BD62]:

$$J^n(x_n) = \max_{u_n} \mathbb{E}\left[r(x_n, u_n) + J^{n+1}(f_n(x_n, u_n))\right] \tag{3.3}$$

This formulation comprises the following elements:

- A training session consists then in a sequence of N tracks.
- The stage of the system is denoted by $n = 0, \ldots, N$.
- The state of the system at stage n, x_n, containing all the required information for the CN to take a decision at each stage (*e.g.*, the *HR*).
- The control applied at each stage, u_n, is the track selected (each one with a different difficulty level). In this case $u_n = \{hard, easy\}$.
- The reward obtained at each stage, $r(x_n, u_n)$, is a function of the state and the control selected. Whenever a track is traversed in $\widehat{HR}$ there must be a positive reward added ($+1$ in our implementation). Otherwise, there is no reward.
- The value function, $J^n(x_n)$, is the maximum expected reward that can be captured from stage n to stage N, *i.e.* if the optimal decisions u_n are taken from the current stage to the last one.

The recursive equation (3.3) states that the overall value function at stage n, $J^n(x_n)$, can be computed as the sum of the reward expected in state x_n plus the value function fur the next stage, $n+1$. An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy regarding the state resulting from the first decision (Bellmans's principle of optimality).

In our particular problem, the system state x_n comprises the following information:

- The class of the user's HR in state x_n (one out of the m possible classes), which we denote as hr_{x_n}.
- The track selected at state x_n (one out of t possible tracks), track_{x_n}.

- The number of loops remaining at each HR class to complete the training, which we denote as $R^1_{x_n}, R^2_{x_n}, \ldots, R^m_{x_n}$.

Note that the control applied to each stage, u_n, is also included in the state of the system. It is necessary to compute the value function J.

In the general multi-goal case, the target is to perform R^1 tracks in HR^1, R^2 in HR^2 and so forth. Hence $N = R^1 + \cdots + R^m$. Whenever a track is traversed in HR^i there must be a positive reward added to the value function if the number of remaining loops in HR^i is positive. Otherwise, there is no reward. Therefore in the general problem, x_n comprises three elements: (1) the athlete's HR, which we denote as hr_{x_n}, (2) the selected track and (3) the number of loops remaining of each HR class to accomplish the objective.

Therefore, the reward function for the problem is defined as:

$$r_n\left(x_n, u_n\right) = \begin{cases} 1 & \text{if } R^{hr_{x_n}} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

Note that, in this case, the reward does not depend either on the selected control or the stage, but only on the state.

The dynamic program can now be formulated. It basically consists computing the value function $J^n\left(x_n\right)$ recursively as the sum of the reward expected in state x_n plus the value function for the next stage, $n+1$.

$$J^n\left(x_n\right) = r\left(x_n\right) + \max_{u_n} \mathbb{E}\left[J^{n+1}\left(f_n\left(x_n, u_n\right)\right)\right] \ , \ \forall\, n = 0 \ldots N \tag{3.5}$$

To solve (3.5), the transition mapping $f\left(\cdot\right)$ must be known. However, the transition from one state to another is not deterministic, due to the random nature of HR in outdoor sports. To overcome this limitation, we can model our system as a *Markov decision process* (MDP). That is, the decisions must be made in a Markov context where the system stochastically evolves from one state to the next one. In a MDP it is assumed that the probabilities $p^{(n)}_{ij}\left(u_n\right)$ of going from state i to state j when control u_n is applied at stage n are known, leading to transition probability matrices $\mathbf{P}^{(n)}\left(u_n\right)$. That is we have to compute:

$$p^{(n)}_{ij}\left(u_n\right) = \Pr\left[x_{i\to j} \mid u_n = \text{track}_j\right] \tag{3.6}$$

where

$$x_{i\to j} = \overbrace{\left(hr_i, \text{track}_i, R^1_i, R^2_i, \ldots, R^m_i\right)}^{\text{state}_i} \to \overbrace{\left(hr_j, \text{track}_j, R^1_j, R^2_j, \ldots, R^m_j\right)}^{\text{state}_j}$$

In this case we can rewrite (3.5) as:

$$J^n\left(x_n\right) = r\left(x_n\right) + \max_{u_n}\left(\sum_{\forall j} p^{(n)}_{x_n j}\left(u_n\right) J^{n+1}\left(j\right)\right) \ , \ \forall\, n = 0 \ldots N \tag{3.7}$$

Expressed in matrix form:

$$\mathbf{J^n} = \mathbf{r} + \max_{u_n}\left(\mathbf{P^{(n)}}\left(u_n\right)\mathbf{J^{n+1}}\right) \tag{3.8}$$

Therefore, the problem of knowing $f(\cdot)$ gets reduced to computing the transition matrices of the system. For simplicity, in this first approach we have considered a static transition matrix. That is, the transition matrix does not depend on stage n. In our experiments, we have computed the probability transition matrices by evaluating the frequencies of the transition events experimentally (see Section 3.3.2.1). This is simple since:

$$p_{ij}^{(n)}(u_n) = \begin{cases} \Pr\left[(hr_i, \text{track}_i) \rightarrow (hr_j, \text{track}_j)\right] & , \begin{array}{c} R_i^{hr_j} - R_j^{hr_j} = 1 \text{ and} \\ R_i^k = R_j^k, \forall k \neq hr_j \end{array} \\ 0 & , \text{otherwise} \end{cases} \tag{3.9}$$

There is only needed to compute the transition probability of going from (hr_i, track_i) to (hr_j, track_j).

Finally (3.5) can be written just as following:

$$\mathbf{J^n} = \mathbf{r} + \max_{u_n} \left(\mathbf{P}(u_n)\, \mathbf{J^{n+1}}\right) \tag{3.10}$$

In order to compute the transition matrices of the system we have considered a static transition scheme. That is, the transition probabilities do not depend on stage n. In our experiments, we have computed the probability transition matrices by evaluating the frequencies of the transition events experimentally (see Section 3.3.2). Therefore, we only need to compute the transition probability by estimating the relative frequencies of going from going from each combination of track and HR (track_i, HR^i) to each subsequent combination of track and HR (track_j, HR^j).

Since N stages are considered, the dynamic program has a finite horizon and the value function for the last stage is $\mathbf{J}^N = \mathbf{r}$. Besides, since in real training the athletes start with a warm-up (WU) period, no reward is considered for the first stage, that is, $\mathbf{r}_0 = \mathbf{0}$.

If the initial state x_0 is known, the value function is evaluated just as $J^0(x_0)$. With the reward function as previously defined, this value can be interpreted as the average number of tracks that the athlete will perform in the correct HR range if the optimal control is applied. The closer the value to N, the better the training. In the next section we evaluate this model with a simple experiment, and compare the results with some simple non-optimal policies.

3.3.2 MDP support in the prototype

As described previously, the goal was to keep the planned activity levels for athletes. The operation of the decision engine is shown in Figure 3.7. The system requires information from previous training sessions to build the transition matrices. Then, those data are used as an input to the MDP block, which selects the optimal policy using the table that corresponds to current environmental conditions. Note that the optimal policy is in fact a mapping between all the system states and their associate best controls (the tracks). This representation of the optimal policy is often denoted as *lookup table* in MDP-related literature. At each decision epoch, the *Track Selection* block retrieves from the lookup table the track that the runner should follow according to the athlete's HR, the last track selected, and the remaining tracks in each HR class ($i.e.$ the parameters determining the current state).

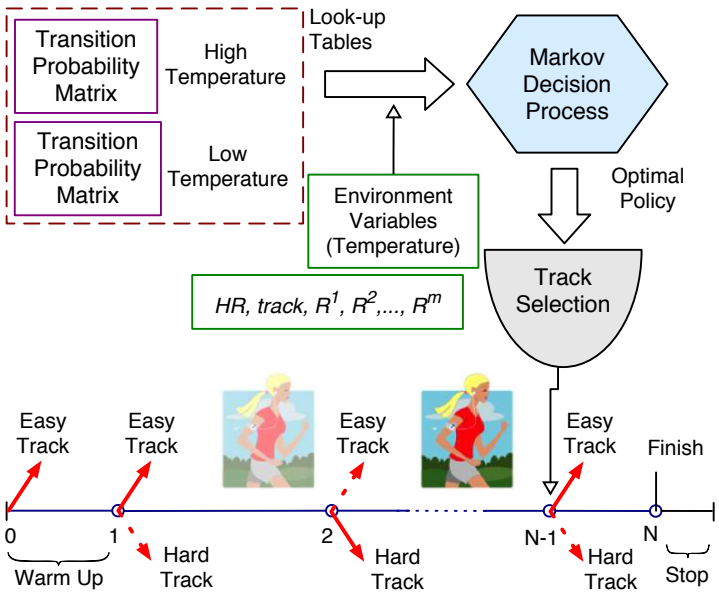


Figure 3.7: Decision process scheme based on MDP.

3.3.2.1 Experiments: computation of the transition matrices

The outdoor network prototype we deployed has ten INs in the hard sector of the training area, and eight in the easy one (see Figure 2.6(b)). The CN was placed in the junction of both loops. Thus, regarding the model described in Section 3.3.1, the number of control decisions is $t = 2$, that is, $u_n = \{Hard, Easy\}$, independently of the stage n. The optimal policy must decide between the hard or the easy track at each stage (see Figure 2.6(a)).

In our experiments (see Section 3.1), we have computed the probability transition matrices by evaluating the relative frequencies of going from each combination of track and HR, (track_i, HR^i), to each subsequent combination of track and HR, (track_j, HR^j). These matrices have been obtained from the data captured in the experiments described in the introductory section, to select the optimal policies for the athlete. Indeed, three matrices were obtained:

- Low Temperature matrix, which only accounts for transitions in low temperature (*i.e.*, average track temperature below 25 Celsius degrees).
- High Temperature matrix, which only accounts for transitions in high temperature (*i.e.*, average track temperature higher than 25 Celsius degrees).
- Absolute matrix, computed using information from all transitions, regardless of the temperature.

Besides, to reduce the overall number of states of the system, which may render the algorithm computationally infeasible, only three HR classes have been used (see Section 3.1):

- HR^1 comprising Low, Moderate and Weight Control HR classes.

- HR^2 which corresponds to Aerobic HR class.
- HR^3 comprising Anaerobic, VO2 MAX and High HR classes.

3.3.3 Evaluating the method on training programs

The performance of the dynamic program can be computed applying the transition matrices of the experiments in the method of Section 3.3.1. Recall from Section 3.3.1 that the value function $J^n(x_n)$ represents the number of tracks that are successfully performed in some of the requested HR classes. The corresponding results are described in this section. Moreover, the optimal policy is compared to the following ones:

- Easy policy, *i.e.* selecting always the "easy" track of the circuit.
- Hard policy, *i.e.* selecting always the "hard" track of the circuit.
- Worst policy, easily computed substituting the *max* operator for the *min* operator in (3.10).

The initial state is always $x_0 = (HR^1, \text{Easy}, R^1, R^2, R^3)$, where R^1, R^2, R^3 denote the training configuration. Hence, the value function of the whole test is computed from (3.10) as $J^0(x_0)$. Let us remark that this performance metric is scalar, since all athletes depart from the same initial state. In the next sections, different training programs are evaluated. In addition, note that in all tests the temperature is considered constant during all the training.

3.3.3.1 Single-goal optimization

In this case the training program to fulfill is $(R^1 = 0, R^2 = i, R^3 = 0)$ that is, performing all tracks in the aerobic (cardio-training) regime. Figure 3.8 shows the results for $i = 1, \ldots, 10$ using the absolute matrix. As expected, the optimal policy achieves better performance than non-optimal ones. In addition, Figure 3.8 depicts a comparison of the optimal policy using either a specific matrix (high and low temperature) or the absolute one. If the specific ones are used, the value function improves significantly, especially in high temperature conditions.

This experiment clearly demonstrates that using environmental information is highly advisable.

3.3.3.2 Multi-goal optimization

In addition, two multi-goal training programs have been studied:

- $(R^1 = 10 - i, R^2 = i, R^3 = 0)$. In this case, 10 loops must be performed with a different distribution in two HR classes. Figure 3.9 shows the results. The trends are similar to those of the single-goal experiment, yet with a larger difference between the optimal policy and non-optimal ones.

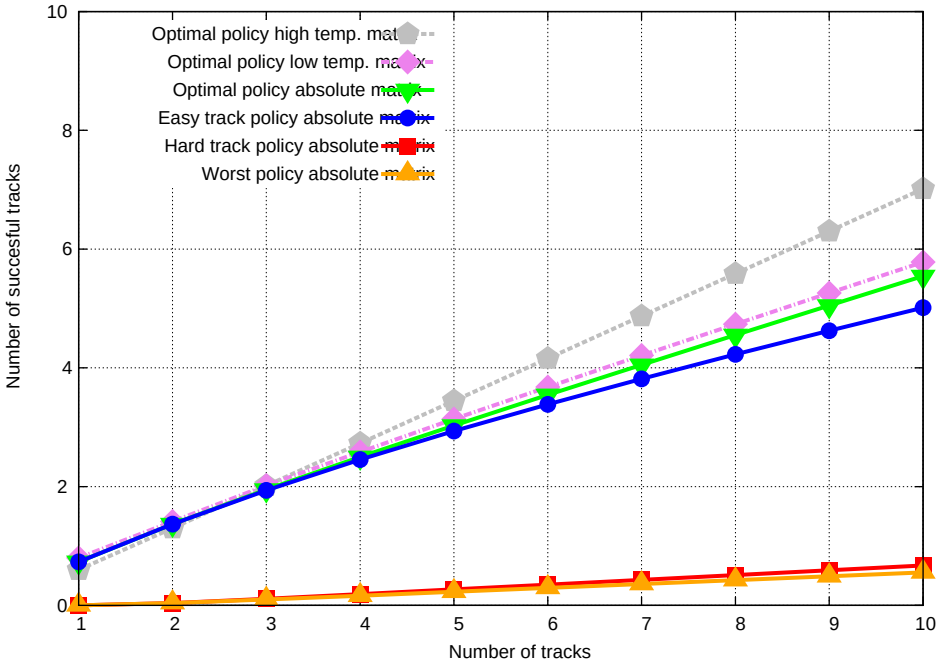


Figure 3.8: Single-goal $(0, i, 0)$ training. Comparison of policies. Optimal policy results with specific and absolute matrices.

- $(R^1 = 7 - i, R^2 = i, R^3 = 3)$. In this case, 7 loops are always performed with a HR distribution across the three classes. Figure 3.10 shows the results. Again, similar trends as in previous experiments have been obtained, confirming the suitability of our proposal.

3.4 Summary

Decision support based on k-NN, and (m, s)-spline interpolation classifiers has been tested for single-step decision-making. Besides, another set of multi-step techniques has been introduced based on dynamic programming. In this case, the goal was to maximize the performance for all future training steps, rather than considering only a horizon with a single decision step. Although not perfect, the classification methods achieves a ratio success in the range from 70% (k-NN) to 85% (splines interpolation). DP demonstrates a 70% match in an homogeneous training program. Note that this is similar to the k-NN classification ratio however, both results are not directly comparable, since k-NN success ratio includes the out-of-range events, whereas DP do not. DP results are therefore more impressive. Furthermore, environmental data improves outcome in all methodologies studied.

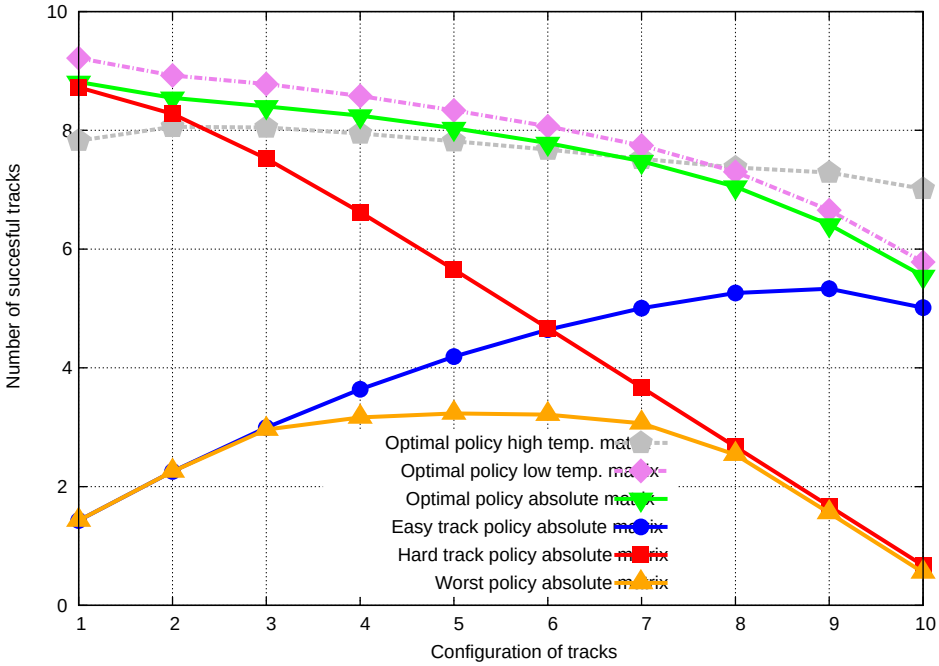


Figure 3.9: Multi-goal $(10-i, i, 0)$ training. Comparison of policies. Optimal policy results with specific and absolute matrices.

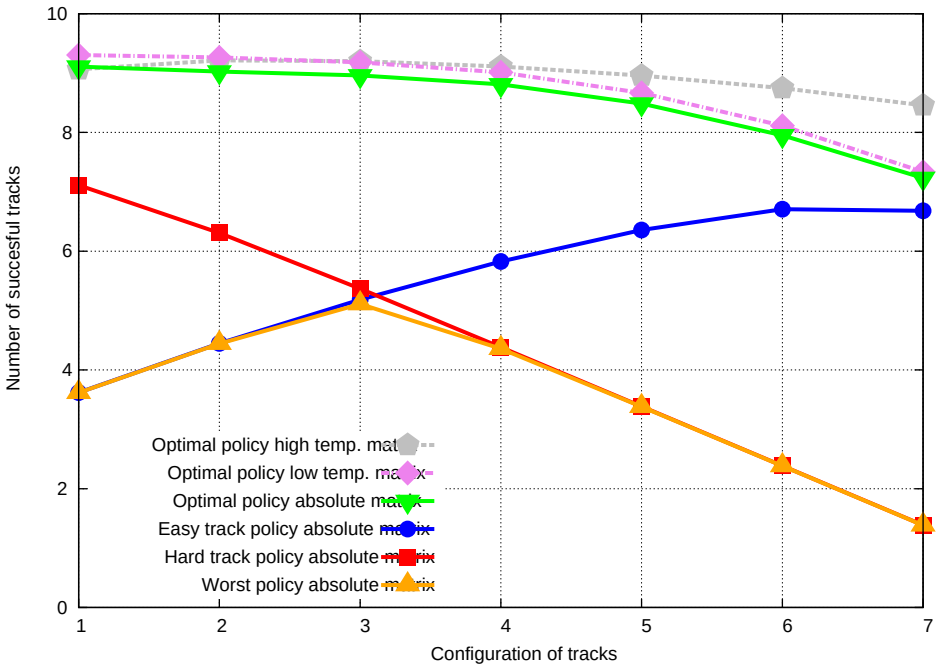


Figure 3.10: Multi-goal $(7-i, i, 3)$ training. Comparison of policies. Optimal policy results with specific and absolute matrices.

Chapter 4

Deterministic placement of AmI networks

As we stated in the introductory chapter providing suitable data for the AmI decision-making procedures is mandatory. These data are required by reasoning processes to infer correct hypothesis about the behavior of the system. Examples of this kind where provided in the previous chapter, where a knowledge base, together with real-time environment and user data allows the AmI prototype to make accurate real-time decisions.

Moreover, the sensors not only monitor their environment, but interact among themselves conveying data to the nodes in charge of decisions. Therefore, the designer must carefully plan their positions, to allow the accomplishment of both tasks. The goal of optimal sensor node deployment goal is to provide criteria to perform both environmental monitoring and connectivity tasks in the best way possible.

As described in the first chapter, placement could be either deterministic (the position of the nodes can be accurately selected), or random (there is only partial control on the position of the nodes). The next chapter addresses issues related to random deployments. This chapter focuses on the study of two deterministic deployment strategies. These cases are modeled as optimization problems composed by one and two objective functions, respectively. The goal is to find the positions where the maximum sensing quality is achieved. Minimizing cost is also an objective in the second example of deterministic deployment analyzed. In both cases network connectivity is a constraint.

Following the AmI environments analyzed in the previous chapters we have considered outdoor target areas for the deployment. These scenarios are more challenging than their indoor counterparts. The results from the first case study are directly applicable to the sport AmI environment addressed in this thesis, but not constrained to it, and may be useful for many AmI systems planning. In this case the optimization methodologies are based on the *ant colony optimization for continuous domains* ($ACO_{\mathbb{R}}$) metaheuristic

In the second deployment example, results apply to the goal of locating a gunshot, in real-time, in order to detect poaching. In this case, the optimization methodology includes a combination of a derivative-free descent method with a Pareto frontier. The Pareto frontier represents the equilibrium between network cost and information quality. This methodology can be applied where cost is also a key factor in network design.

4.1 Introduction

The quality of WSNs deployment is commonly related to the coverage of the network in the region where service is offered. Coverage objectives change with the application domain. In ambient monitoring and surveillance applications, for example, the objective is to maximize the sensing area, but achieving suitable coverage may be complicated due to the size of the target area and due to the number of nodes involved. Therefore, node positions must be carefully planned before deployment. Connectivity between sensor nodes is also a major issue since information has to be conveyed through a multi-hop path to the *sink* node; otherwise data will be lost.

The quality of sensed data is not a uniform metric, although most previous works [LT04, MKPS05, LDWS08] treat it as such. In real applications, sensing locations where valuable events occur with a high probability must be selected with greater priority. Examples in biological or geological surveys are areas where animals rest or feed [PSM+04], or areas with registered earth activity [PLSS08]. For military-related applications, it may be important to capture information on troops and vehicle location and tracking [HkS+09]. Thus, bridges, roads, or mountain passes may be critical. In a nuclear power plant, for example, it is essential to control temperatures and to monitor the vibration of the reactor and radioactivity levels in safety zones [LWS04]. Early warning in case of hazardous levels on these parameters before an eventual failure is critical.

Sensors placement should then allow the best event detection or collect the most of this important information. Determining the optimal placement of sensors to maximize sensed data importance is crucial. Therefore, we provide a quantitative metric for the quality of the network which is related to the relevance of the data captured rather than to network coverage.

4.1.1 Outline of this chapter

As stated above, we study two different sensor positioning problems: single-objective and dual-objective network deployment problems. In both problems we assume a two-dimensional target area in which a non-uniform distribution of data importance is defined.

In the first part of the chapter (Section 4.2) we propose a new optimization scheme which selects the optimal node positions in order to maximize the quality of the data captured by the network. Thus, we consider only one objective to optimize when positioning nodes. For this problem, we propose a combinatorial optimization problem where the positions of nodes are variables, leading to an $\mathcal{NP}$-hard problem, which must be tackled by means of heuristics to obtain a solution within a bounded time.

Among those algorithms we have selected (ACO) [Dor92]. ACO has already proved its worthiness solving a wide set of different combinatorial optimization problems (*i.e.* problems applied to discrete domains only). Nevertheless, to address our optimization problem we have utilized a recent research [SD08] that extends ACO to handle continuous domains. This algorithm is denoted as $\text{ACO}_{\mathbb{R}}$. To the best of our knowledge, the present work is the first aimed to solve a sensor location problem using this algorithm. We demonstrate the strength of $\text{ACO}_{\mathbb{R}}$ by comparing our results with those obtained using a *four-directional placement* (FDP) algorithm. FDP is a simple heuristic which uses

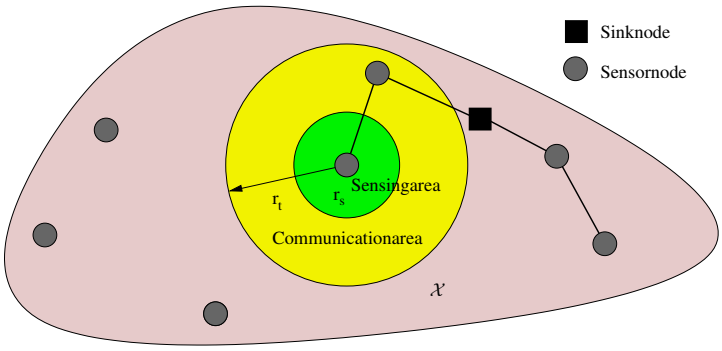


Figure 4.1: Connectivity and sensing parameters.

a grid placed over target area to determine candidate points and selects the best according to importance.

In the second part of the chapter (Section 4.3) we develop the network design and planning of an acoustic sensor network whose goal consists of locating gunshots (to detect illegal hunting) in national parks. Since the areas under surveillance are wide, and electric power is scarcely available in them, it is necessary to maximize detection coverage and minimize system cost at a time. Therefore, sensor network planning has two opposite targets, coverage and cost. We model planning as an unconstrained problem with two objective functions. In addition to the optimal deployment problem it is necessary to provide the system with some procedure to locate the point where the shot has been produced. This is also developed in Section 4.3.

4.2 Single-objective deployment problem

4.2.1 Mathematical formulation

Our model comprises a set of N nodes (including one base/sink node per network). This network must be deployed in a non-homogeneous target area, such that importance (quality) captured will be maximized. The planning is subject to communication constraints, since nodes must have a path (either directly or via a multi-hop route) to the base node. Figure 4.1 illustrates the main concepts of the model.

Let $\mathcal{X} \subseteq \mathbb{R}^2$ be the target area. There exists an *importance* mapping (representing traffic, sensing information, etc.) associated with every $x \in \mathcal{X}$ which is a real and continuous *function* $\alpha : \mathcal{X} \to \mathbb{R}_0^+$. Let $\{x_i\}_{i=1,\ldots,N}$ be the set of each node i position. Besides, the following considerations have been established:

- The hardware of the nodes is homogeneous, *i.e.* it is of the same type and has the same communication/sensing capabilities.
- The sink node can be any of the N sensor nodes. Without loss of generality, we assume that it is node 1 and therefore x_1 is its position.
- The *sensing range* r_s is the longest distance between the node and the locations it

is able to sense. It is assumed that a node is able to monitor the whole area within its sensing range. Therefore the information captured by sensor i is:

$$\int_{B(x_i,r_s)} \alpha(x)\mathrm{d}x \tag{4.1}$$

where $B(x,r)$ is the open ball in $\mathbb{R}^2$ centered in x with radius r.

- The *transmission range* r_t is the longest distance between two mutually communicating nodes. Let us term nodes able to transmit sensed data to the sink as *active nodes*, and let $A^\star$ denote this set of nodes.

- If sensing areas of active nodes overlap, the information captured does not increase. Thus, the open set describing the sensed area is

$$\mathcal{B} = \left(\bigcup_{i \in A^\star} B(x_i, r_s) \right) \bigcap \mathcal{X} \tag{4.2}$$

Therefore, we can state that our objective is to find the node positions x_i for $i = 1, \ldots, N$ over the target area such that:

$$\max_{\substack{x_i \\ i=1,\ldots,N}} \int_{\mathcal{B}} \alpha(x)\mathrm{d}x \tag{4.3}$$

which measures the importance of the data captured by active nodes.

Optimal node placement formulated in (4.3) is a very challenging problem. In fact, most of these problem formulations (*e.g.* [EHPM05, PPKS06]) have been proved to be $\mathcal{NP}$-hard. In other words, it is strongly believed that it is not possible to find an efficient (*i.e.* polynomial time) algorithm to solve them optimally. Usually these problems are addressed by means of heuristics which allow to find suboptimal (*i.e.* approximate) solutions in a reasonable amount of time. Several heuristics have been proposed to find suitable sub-optimal solutions for the node placement problem [DC03, PCH+05]. Next section shows our optimization model approach based on $\mathrm{ACO}_{\mathbb{R}}$ metaheuristic for continuous domains [SD08].

4.2.2 $\mathrm{ACO}_{\mathbb{R}}$ adaptation to the deployment problem

ACO methodology is essentially inspired in the foraging behavior of real ants. Real ants are social insects that coordinate their activities via *stigmergy*, a form of indirect communication mediated by modifying the environment. Once they have found food, individual ants mark good paths from food sources to their nest by depositing *pheromone* (*i.e.* chemical substance) on the ground. The amount of pheromone deposited may vary with the quantity and quality of the food found and serves to guide other forager ants. This pheromone trail reinforcing mechanism is utilized by ACO metaheuristic to find approximate solutions to discrete optimization problems. In fact, the search process for optimal solutions involves reinforcing the pheromone of good solutions and weakening that of bad

```
while termination conditions not met do
  ScheduleActivities
     ConstructAntsSolution
     PheromoneUpdate
     DeamonActions % optional
  end ScheduleActivities
end while
```

Figure 4.2: Ant Colony Optimization metaheuristic.

ones. This mechanism helps to guide ants towards better solutions. $ACO_{\mathbb{R}}$ optimization methodology conserves the ACO fundaments. Informally, the ACO and $ACO_{\mathbb{R}}$ optimization metaheuristics are shown in Figure 4.2. Differences arise mainly in pheromone representation and in the handling of continuous variables when constructing new solutions. A detailed description of ACO and $ACO_{\mathbb{R}}$ are provided in [SD08] and [Dor92], respectively.

In the ACO metaheuristic, artificial ants carry out an incremental construction of solutions. For each step each ant chooses a solution component from the set of *neighboring* (*i.e.* feasible) components, which constitutes the ant search space. The selection is biased by the amount of pheromone previously deposited on each feasible component. Since the search space is discrete, the probability distribution is a discrete random variable, which is represented by its *mass function*. On the contrary, in $ACO_{\mathbb{R}}$ the search space is continuous. Thus, to select a feasible component it is necessary to obtain samples of a continuous random variable (described by its *probability distribution function* - pdf) representing the new components.

On the other hand, $ACO_{\mathbb{R}}$ uses a constructive pheromone approach. The algorithm keeps track of a certain number, K, of *good* solutions (the best ranked) in a *solutions archive* T (see Figure 4.3). Each row l in T represents an instance of the solution of our optimization problem, which contains the placement coordinates (*i.e.* the component) of the N nodes. At each $ACO_{\mathbb{R}}$ step, the components of the archive T are used to construct new solutions.

Thus, our placement optimization algorithm begins by initializing the solutions archive with $K(\geq N)$ solutions generated randomly. Each solution (*i.e.* each row in archive T) is composed of N node coordinates $x_i, i = 1, \ldots, N$. These solutions are sorted in T according to the objective function value $f(\cdot)$. Function f is the captured importance and is computed with (4.3). Vector $\mathbf{w} = (w_1, \ldots, w_K)^t$ ranks each of the K solutions depending on the position in T (*i.e.* on the row number) with the following Gaussian function:

$$w_l = \frac{1}{qK\sqrt{2\pi}} \, \mathrm{e}^{-\frac{(l-1)^2}{2(qK)^2}} \tag{4.4}$$

Where q is an operational parameter of the algorithm. Note that qK is the standard deviation of (4.4). Since archive T represents the ant search space, the choice of parameter q modulates the chance of selecting zones within this space. For example, search zones represented by the best-ranked solutions can be given more or less priority by adjusting the q parameter.

Once the archive of solutions T is initialized, the optimization algorithm follows the

s_1	x_{11}	x_{12}	$\cdots$	x_{1i}	$\cdots$	x_{1N}	$f(s_1)$	w_1
s_2	x_{21}	x_{22}	$\cdots$	x_{2i}	$\cdots$	x_{2N}	$f(s_2)$	w_2
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\rightarrow s_l$	x_{l1}	x_{l2}	$\cdots$	x_{li}	$\cdots$	x_{lN}	$f(s_l)$	w_l
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_K	x_{K1}	x_{K1}	$\cdots$	x_{Ki}	$\cdots$	x_{KN}	$f(s_K)$	w_K
	G_1	G_2		G_i		G_N		

Figure 4.3: Solutions table used by $\text{ACO}_{\mathbb{R}}$ in the placement optimization problem. The solutions are sorted in the archive according to their quality (captured importance f). Each solution has an associated weight (w_i) that is proportional to the quality of the solution.

algorithm described in Figure 4.2. Artificial ants are also established to compute new placement solutions for testing. At each iteration, each ant is in charge of constructing a new solution by using the pheromone values represented by the solutions stored in archive T.

To generate a new position for a node i, first a new random variable describing its possible values is obtained. In this random variable, the positions with a higher level of pheromone are more likely. The pdf of this new random variable is generated as the weighted sum of pdfs:

$$\forall x \in \mathcal{X},\, G_i(x) = \sum_{1 \leq l \leq K} w_l g_{li}(x, x_{li}, \sigma_{li}) = \sum_{1 \leq l \leq K} w_l \frac{1}{2\pi\sigma_{li}^2} \mathrm{e}^{-\frac{|x - x_{li}|^2}{2\sigma_{li}^2}} \tag{4.5}$$

Where $g_l i$ denote the pdf of the Gaussian distribution associated to the position of node i in solution l. The value of σ_{li} is the average distance from the component i of the selected solution l, to the same component of the other solutions in T. It is given by:

$$\sigma_{li} = \xi \sum_{1 \leq e \leq K} \frac{|x_{ei} - x_{li}|}{K - 1} \tag{4.6}$$

The parameter $\xi > 0$ operates in an analogous way to the pheromone evaporation rate in regular ACO and allows the algorithm convergence speed to be controlled. The higher the values of ξ the less biased the solutions will be. In other words, ants will move more likely towards search zones not already evaluated by the algorithm.

Summarizing, each of the $l = 1, \ldots, K$ solutions in T contributes to the computation of G_i with a Gaussian distribution centered at x_{li}. Note how this scheme allows generating new solutions from previously inspected ones (mimicking ant behavior). Then, the artificial ant obtains a sample φ_i of G_i as the new position for the node i.

In practice, the sampling of G_i can be performed in two phases using a simple method. First, the ant will select one previous solution l (*i.e.* a row in T). The probability p_l of

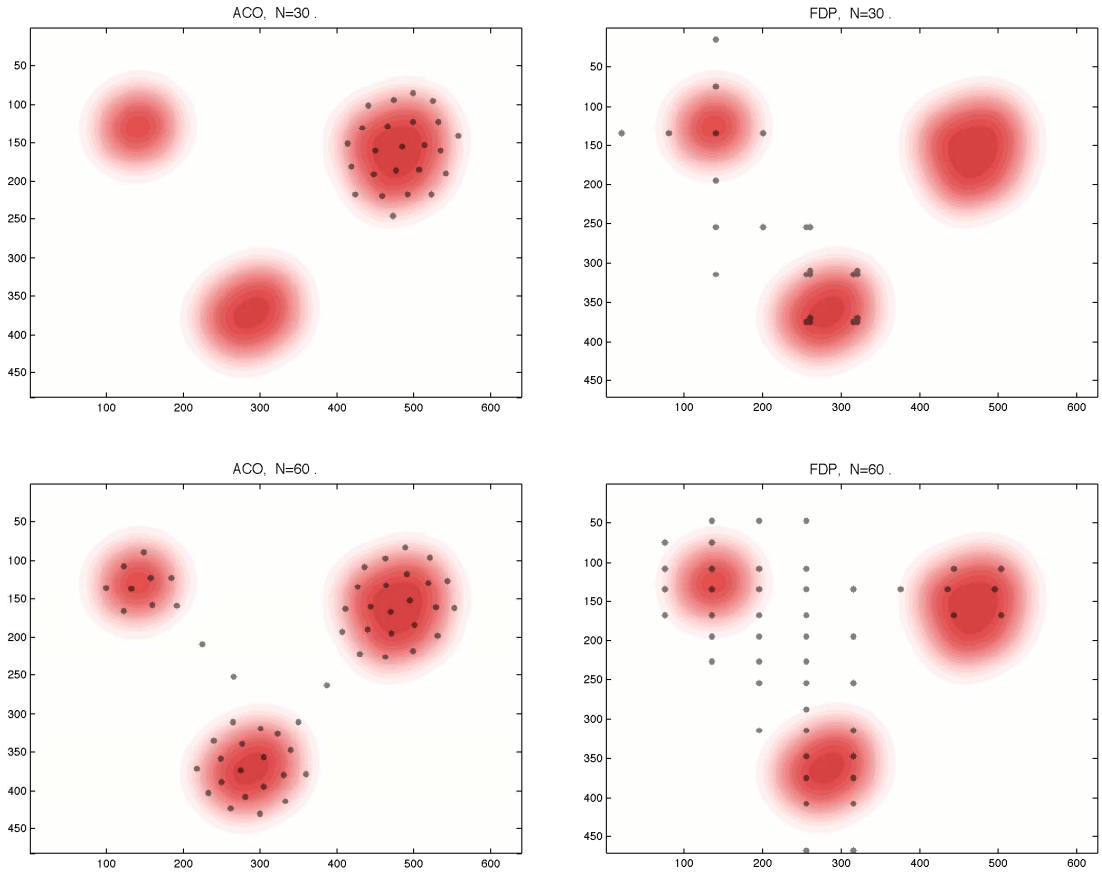


Figure 4.4: Node placement examples for different number of nodes in a convex scenario. $\text{ACO}_{\mathbb{R}}$ (left) and FDP (right).

choosing solution l is given by:

$$p_l = \frac{w_l}{\sum_{1 \leq r \leq K} w_r} \tag{4.7}$$

Second, the ant will generate a sample φ_{li} from the corresponding Gaussian distribution associated to row l. This sample (representing the position for node i in the new solution) is selected as φ_i. For constructing the whole new solution, the artificial ants just generate samples for the random variable of each node i.

Before next iteration, the algorithm performs a partial renewal of archive T. This mechanism permits to modify the search space to guide the ants in the search process more effectively. The pheromone update procedure consists, thus, of adding the set newly generated solutions to archive T, reranking all solutions, and eliminating the same number of worst-ranked solutions. This process also finished with K rows in T.

Finally, the algorithm ends when no improvements for the highest ranked solution are found after a given number of iterations (*maxiter*).

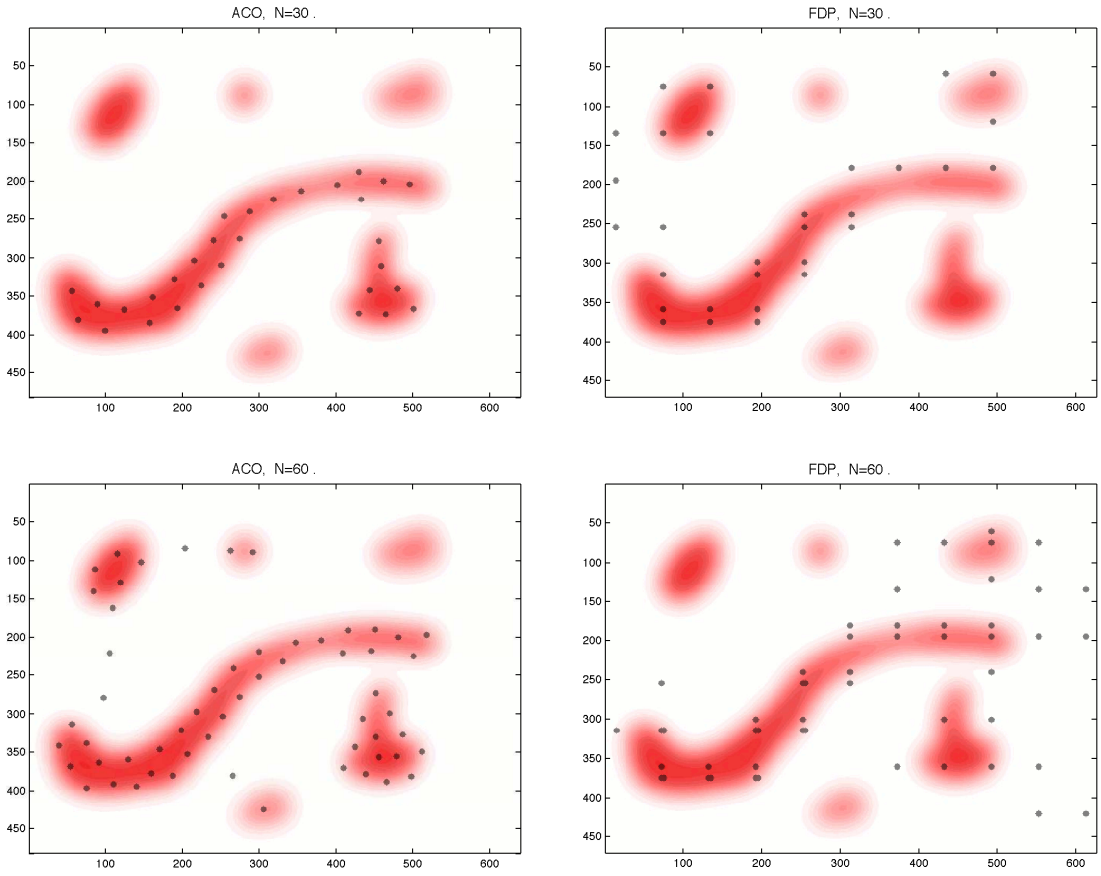


Figure 4.5: Node placement examples for different number of nodes in a non-convex scenario. $ACO_{\mathbb{R}}$ (left) and FDP (right).

4.2.3 Heuristic evaluation

We tested our $ACO_{\mathbb{R}}$-based algorithm in two scenarios: (a) a *convex* scenario with three large convex regions, and (b) a *non-convex* with six separate regions, including one large non-convex set crossing the scenario. Sample deployments are shown in Figures 4.4 and 4.5, respectively. Each scenario is a bitmap image with a resolution of 640×482 pixels. Each pixel in the image has an associated value, ranging from 0 (least importance) to 255 (maximum importance). Figures 4.4 and 4.5 show different levels of importance graded on a red scale (light red $\rightarrow$ not very important, dark red $\rightarrow$ very important).

Based on adjustment tests, the optimal number of ants was set to 10 and the maximum number of iterations ($maxiter$) to 4500. $ACO_{\mathbb{R}}$ parameters q and ξ were set to those determined in [SD08], *i.e.* 10^{-4} and 0.85 respectively.

In order to study the performance of our proposal, we contrasted our results with those obtained using the FDP heuristic in which a grid was placed over the target area, with r_t space between vertices. The FDP is an iterative algorithm. At each step it selects a new point. This point is the most important point adjacent to the previously selected point and cannot have been previously selected. If there is a tie, FDP chooses at random. At

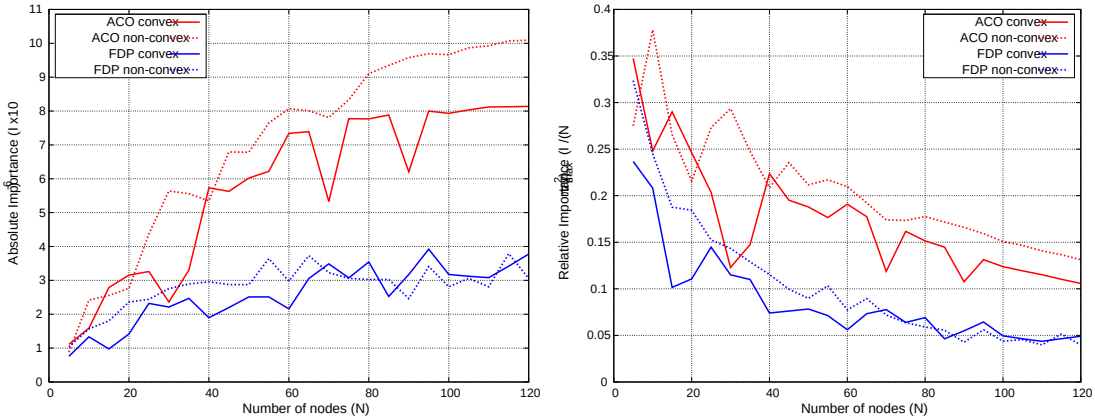


Figure 4.6: Performance comparison of $ACO_{\mathbb{R}}$ with FDP heuristic.

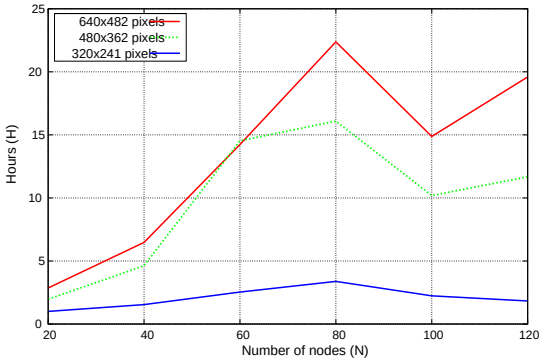


Figure 4.7: Time consumed as a function of number of nodes and scenario size.

each iteration, the points that are evaluated but not selected are kept in a sorted table (observed points table) in descending order of the importance (determined by importance function value). In case of a dead end, the algorithm goes back to a point not previously selected by choosing/removing the first point of the observed points table (thus ensuring connectivity).

In the two scenarios (convex and non-convex) we spread $N = 5i$, $i = 1, \ldots, 24$, sensors with a coverage radius of $r_s = 20$ units and a transmission range of $r_t = 60$ units (a total of 48 deployment simulations). The numerical results are shown in Figure 4.6 and a sample of the graphical results in Figures 4.4 and 4.5. Figure 4.6 shows absolute and relative captured importance. Relative importance (I_{rel}) is computed by dividing total importance (I) by the ideal importance that can be captured with N nodes. Thus, $I_{rel} = I/(N\pi r_s^2 v_{max})$, where $v_{max} = 255$, the maximum importance assigned to a pixel in the map.

The results shown in Figure 4.6 demonstrate that $ACO_{\mathbb{R}}$ outperforms the FDP heuristic even in scenarios with few nodes.

Figure 4.7 depicts the evolution of the algorithm computing time with respect to the number of nodes and the map size in the convex scenario. As expected, both parameters (nodes and size) increase the computing time, but not in an exponential fashion.

Figure 4.8: Landscape of the Cabañeros National Park.

In summary, the results show the feasibility of the $ACO_{\mathbb{R}}$ approach for node placement. We demonstrate how it can be easily adapted to a connectivity constrained problem and how solutions are overwhelmingly better than simple heuristics such as FDP.

4.3 Two-objective deployment problem

4.3.1 Introduction to gunshot detection and network planning

The Spanish SEPRONA [Gua] agency fights against poaching with considerable success. However, the problem remains relevant. Since 2007, SEPRONA has detained over 150 people for hunting felonies [ELP, New]. Consequently, we have proposed the DiANa project, *Detección de caza furtIva con Armas de fuego en parques NAcionales* (Detection of illegal hunting with gunfires in national parks), to automatically detect and locate gunshots, which is endorsed by Cabañeros National Park [Spa].

The DiANa system consists of a network of acoustic sensors that locate gunshots in wide open area extensions. The system consists of a network of acoustic sensors that locate gunshots by hyperbolic multilateration estimation. The differences in sound time arrivals allow to compute a low error estimator of gunshot location. Among the methods of detection (see Section 1.3.4) we have utilized the *time difference of arrival* (TDoA) methodology. The TDoA technique exploits the relationship between distance and transmission time when the propagation speed is known. Once the time delays are calculated, they are processed in order to estimate the location of the source. Due to the distances between the deployed sensors (in the order of hundreds of meters), and the smooth landscape in Cabañeros National Park (Figure 4.8), we have assumed a two-dimensional scenario.

TDoA-based ranging techniques require accurate clock synchronization. Every sensor knows its own position exactly, and it records the arrival time of the sound event. So, the sensor clocks must be as tightly synchronized as possible, using dedicated time synchro-

nization algorithms. Since an acoustic signal location in open spaces network must scale to large sizes, it is necessary to minimize the number of exchanged messages to attain convergence, keeping energy consumption at reasonable levels if the electric grid is not available.

Selecting a synchronization schema for real applications like ours is not easy. The best known synchronization schemas implement network mechanisms to adjust all local clocks to the same value. This is achieved by exchanging time stamps between node pairs. The more frequent the exchanges, the higher the time accuracy. Two representative examples are *reference broadcast synchronization* (RBS) [EGE02] and the *timing-sync protocol for sensor networks* (TSPN) [GKS03]. We discarded GPS receivers for their high cost. The Section 4.3.2.3 present a new ad-hoc flood method to set the clock times in every node in the network to the same value. In this method, the nodes do not exchange synchronization messages, and thus they save power. Once a node detects, for example a gunshot, the time of the event is transmitted to the sink node through a previously generated path. The method performs a cooperative backward time adjustment, so that every node along the path to the sink is able to estimate the event time of the node that receives the time stamp.

Regarding network planning, since the areas under surveillance are wide, and electric power is seldom available in them, it is necessary to maximize detection coverage and minimize system cost at a time. Therefore, we model sensor network planning as an unconstrained problem with two objective functions. We provide a set of candidate solutions of importance by combining a derivative-free descent method and a Pareto front approach.

Due to the inherent difficulties exhibited by the thus far formulated models in sensor network planning, several heuristic optimization strategies have been proposed in the literature: variants of simulated annealing [AM94], genetic algorithms [LLL98], gradient descent (when applicable) [SPR96] and others [KU02, UK03].

Some of these approaches (simulated annealing, genetic algorithms and the like) do not guarantee theoretical convergence. Regarding gradient descent methods, the gradient is often unavailable or too costly to compute. Therefore, we have adapted a non-monotone derivative-free optimization technique with guaranteed convergence [GPGCnBR06] to formulate and solve a computationally efficient **optimization model** with a dual objective: maximizing acoustic network coverage and minimizing power infrastructure cost. The results are presented as a Pareto front, revealing solutions that are clearly superior to random seeding.

Our notation is as follows: Lower case Greek letters are scalars, lower case Latin letters are vectors in $\mathbb{R}^2, x_k^j$ is the j-th component of the vector x_k, and $||x||$ is the Euclidean norm. A capital Latin letter, say S, stands for a collection of vectors in $\mathbb{R}^2$ if $S = \{s_1, \ldots, s_p\}$ we also say that $S \in \mathbb{R}^{2p}$; $\tau S = \{\tau s_1, \ldots, \tau s_p\}$, and the sum $Z = S + D$ means that S and D have the same number of elements, say p, and $z_k \in Z$ if *and only if* $\exists k \in \{1, \ldots, p\} \mid z_k = s_k + d_k$. In general the subindex i is the value of an entity (scalar, vector, set, and so forth) at the i-th iteration of an algorithm; for instance $S_i = \{s_{i1}, \ldots, s_{ip}\}$ is a set of p vectors in $\mathbb{R}^2$, at the i-th iteration.

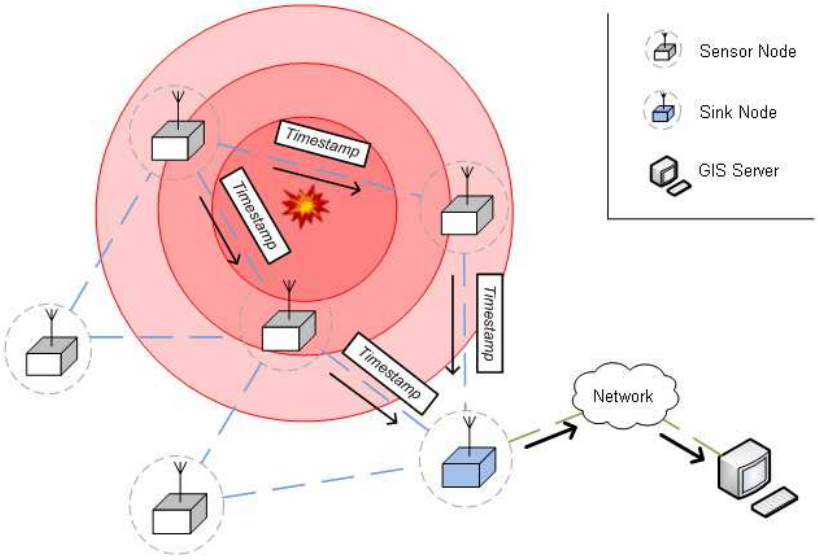


Figure 4.9: Gunshot location architecture.

4.3.2 Gunshot location

In this section we describe the location procedure for acoustic events, which we have implemented for MICAz motes. The goal of our system is gunshot location by means of a sensor network. Location is based on hyperbolic positioning [Pat05]. Hyperbolic positioning requires the sensor clocks to be synchronized, in order to apply the TDoA technique. For this reason we have implemented a synchronization protocol in the MICAz motes. Next, we describe the system architecture, the location method and, finally, the synchronization schema.

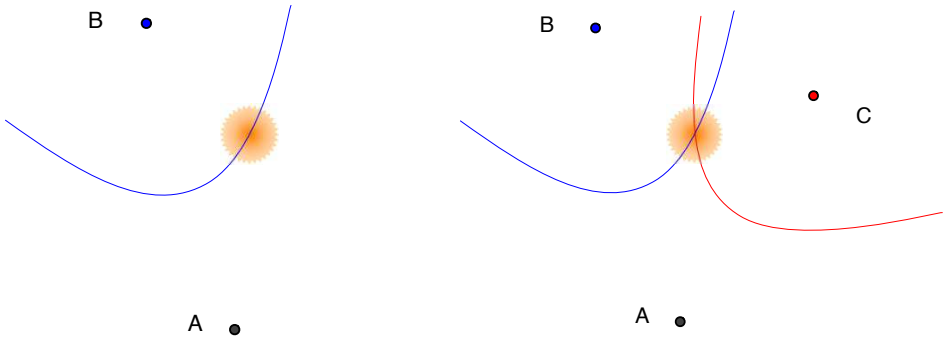
4.3.2.1 System architecture

Figure 4.9 shows the system architecture, with three components:

- Sensor nodes: The sensor nodes in known positions are equipped with the necessary hardware for detection of acoustic events. They can discriminate between *normal* and *shot* segment classes in audio streams. When a sensor node detects a sound event, it transmits a packet with information about the type of sound event and a sound time-stamp to a special node, the sink.

- Sink nodes: Sink nodes collect the packets sent by sensor nodes and deliver them to the GIS server to calculate the position of the sound event. Sink nodes may be sensing nodes as well.

- GIS server: Using the information of the sink nodes, the GIS server estimates the position of the acoustic event by means of a hyperbolic method, described next.



(a) Locus of the TDoA obtained between positions x_A and x_B

(b) Locus of the TDoA obtained between positions x_A and x_B, and between positions x_A and x_C

Figure 4.10: Hyperbolic multilateration of an acoustic signal.

4.3.2.2 Acoustic source location procedure

Since our optimization model (see Section 4.3.3) considers a flat landscape with scattered trees, we have selected a two-dimensional hyperbolic positioning algorithm. Nevertheless, it could be easily extended to three-dimensional location in rough scenarios.

Let us consider an acoustic event that takes place at an unknown position $x \in \mathbb{R}^2$, which we wish to determine. Formally, all sensing node locations $s = (s^1, s^2)$ belong to a well defined *compact* set $X \subset \mathbb{R}^2$. We denote the *Euclidean* distance $\delta(s, x)$ between a sensor location s and another point x in the scenario, $s, x \in X$, by

$$\delta^2(s, x) = \|s - x\|^2. \tag{4.8}$$

Let us also assume that a subset $B \subset X$ of the nodes have detected the event, and that the position of those nodes is known. Let us denote the sound propagation speed as υ. Then, the time of arrival t_b of the event at any node $b \in B$ is:

$$t_b = \frac{1}{\upsilon}\delta(x, s_b) \tag{4.9}$$

and the difference between the arrival times of the event to a pair of nodes s_b and $s_{b'}$ is:

$$\tau_{b-b'} \triangleq t_b - t_{b'} = \frac{1}{\upsilon}\left(\delta(x, s_b) - \delta(x, s_{b'})\right), \quad \forall b, b' \in B \tag{4.10}$$

This expression defines hyperboles, whose intersection determines the source of the shot. Figure 4.3.2.2 shows this process graphically.

For two given receiver locations, s_b and $s_{b'}$, a set of emitter locations would yield the same TDoA measurement. This is the *locus* of possible emitter locations and it describes a hyperbole. If we now consider a receiver at a third location $s_{b''}$, it provides a second TDoA measurement, and, hence, it allows to locate the emitter on a second hyperbole. In general, a set of distances from the source to every sensor pair identifies a set of hyperboles. As

a consequence, the location of the source is the intersection of this set of hyperboles. In Section 4.3.5, our tests show that location accuracy increases with the number of gunshot detection nodes. In most trials, the accuracy in scenarios with three detection nodes or more is sufficient to find the acoustic source (the detailed TDoA location technique is described in [Pat05]).

Although non-linear expression (4.10) has a unique solution if there are enough hyperboles, there is some uncertainty in the calculation since:

- The speed of sound varies depending on altitude, humidity and air temperature. As we have mentioned, multi-path propagation affects the accuracy of acoustic signal detection. Single spread-spectrum techniques such as those in [GE01] largely mitigate it.
- The microphone directionality or polar pattern affects the result.
- The clock drift may drastically vary in time due to environmental temperature and humidity changes. In Section 4.3.2.3 we propose an approach to reduce sensor clock deviations.

Due to these inaccuracies, expression (4.10) may be inconsistent. Nevertheless, low-error estimations are possible. A nonlinear optimization problem can be formulated [ZLL08] to minimize the difference between estimated and real positions. We minimize the square error of the location, defined as the differences between the squares of the theoretical and measured differential arrival times to a reference node $s_{b'}$:

$$x = \arg\min_{x} \sum_{\forall b \in B \setminus s_{b'}} \left((\delta(x, s_b) - \delta(x, s_{b'})) - \upsilon \tau_{b-b'} \right)^2 \tag{4.11}$$

The least square minimization problem in (4.11) is not convex. Thus, standard optimization algorithms, like incremental gradient, are not guaranteed to converge to the global minimum. The initial conditions in an iterative algorithm may lead to a local optimum or a saddle point at the termination, adding imprecision to gunshot location. The required computing power is moderate, and the GIS server (Figure 4.9) can handle the calculations. The solutions can be obtained practically in real-time (see Section 4.3.5). Our synchronization algorithm (see Section 4.3.2.3) also contributes to the location procedure, since it is required for the sink node to compute the TDoA between the detectors. The tests with our reference implementation in MICAz motes reveal that, the more sensors that detect the sound event, the greater the location precision (see Section 4.3.5). This is due to error compensation in arrival time measures. In conclusion, a high density of sensors around a sound event always improves location precision. Therefore, optimal sensor positioning improves the performance of the location system.

4.3.2.3 Synchronization schema

Time synchronization is a fundamental aspect in distributed sensor networks. In the proposed shot detection system, the differences in arrival times can only be computed if the nodes are tightly synchronized. To adjust the clocks, the nodes must exchange messages

indicating the time reference. In fact, time adjustment degrades progressively due to clock drifts, and it is indeed mandatory to readjust it periodically (by sending new messages). However, power consumption is higher if synchronization packets are continuously transmitted. Even if a continuous energy source is available, it may be necessary to extend the network with autonomous nodes at its edges. Therefore, the number of exchanged messages should be as low possible, for a given accuracy goal. For gunshot location purposes, if we assume a maximum error of few meters, the maximum allowed error in time synchronization is in the order of a tenth of a second (d_e/υ, where d_e is the allowed spatial error). For example, three milliseconds of clock drift will cause an estimated error of one meter, relative to the real source position. Thus, fine-grained clock synchronization is mandatory. The tests with our synchronization algorithm show a time accuracy of tenths of microseconds. The implementation has been carried out with TinyOS in MICAz devices.

The synchronization algorithm has two steps:

1. *Level discovery*: This step is similar to the level discovery stage in TPSN [GKS03]. Before the synchronization process takes place, the network has to organize itself as a hierarchical tree, beginning at a root node (in our case we choose the sink). According to the minimum number of hops to the sink, a level is assigned to each node (level 0 to the root). To compute the tree, the process starts at the root, broadcasting a level discovery packet to the nodes at level 0. The nodes that receive this packet are marked as children of the root node, and they set their level to 1. The nodes ignore further level discovery packets with greater or equal level numbers. Then, level 1 nodes broadcast their level discovery packets, and so on. Note that this process also permits to discover optimal communication paths (in number of hops) to the root, and, thus, it is valid for network routing.

2. *Synchronization*: Once the hierarchical network structure is completed, the synchronization process may start. In general, level k nodes synchronize their children (of level $k+1$).

   Besides its own local clock, a sensor node will maintain an estimation of its synchronizer node clock in the upper hierarchical level. The approximation consists of calculating the regression line of those two clocks. Previously, the level k node receives several synchronized time-stamps of level $k-1$ (see Figure 4.11), which are broadcast following the tree structure that was created at the level discovery step. Figure 4.11 shows the regression line used to calculate the *parent* node clock in a level k node. Value α_k represents the clock offset at reference time $t = 0$, and the slope β_k is the rate of change (clock drift) of the local clock.

   Once a node detects a gunshot, it sends the event to its *parent* node in the upper level, according to the parent time clock. After one or more hops, level 0 (sink node) will receive estimations of the detection time that are synchronized with the sink clock, from one or more level 1 nodes. This way, local clock exchanges do not spend power. Since clock drift varies slowly, the regression line must only be calculated every 6 or 8 hours, according to our tests with MICAz motes. Only large temperature variations affect the regression line slope, requiring node re-synchronization.

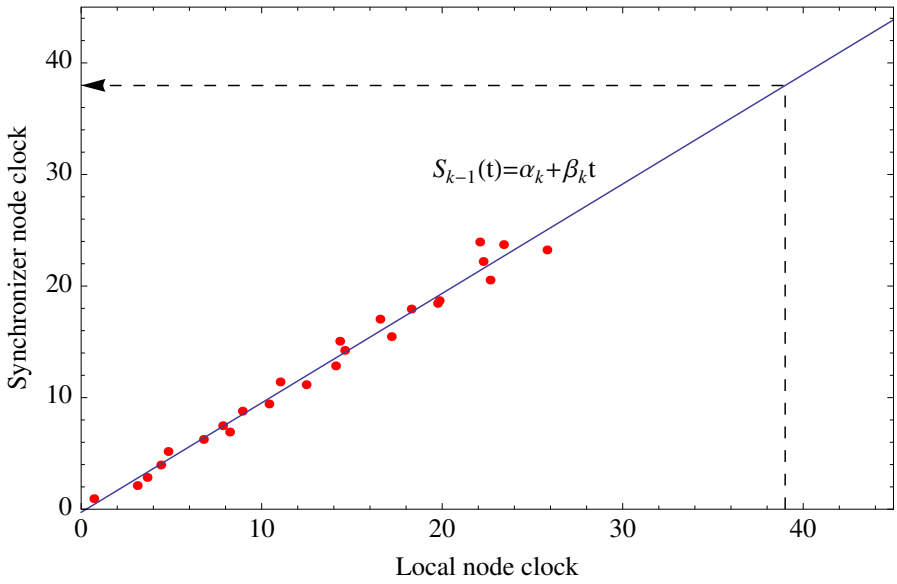


Figure 4.11: Regression line performed in a level k node, with several broadcast time stamps from a level $k-1$ node.

4.3.3 Optimization model

The sound spectrum of a gunshot is dominated by the 130 Hz to 3 kHz frequency range [Sau]. According to [Buc06], at these frequencies there is an extra attenuation of 3 dB each ∼25 m in woods, yielding 12 dB vs. 9 dB in open space. Given the sensitivity threshold of the sensing node, its maximum ranges are 1Km in open space and 750 m in a wood.

We consider a large outdoor flat scenario with open space areas and wood patches. The scenario is crossed by a few power lines. In it, we wish to deploy a given number of fixed sensing nodes, so that the detection coverage is maximum. A point gets covered if it is reachable by a sensing node at least, although this capability increases if more sensing nodes see the point (to achieve source signal location we require a coverage of three nodes at least, as explained in Section 4.3.2.2).

At the same time, we wish to minimize the distance between the sensing nodes and the power lines so that the cost of the power infrastructure is minimal in case the nodes are not autonomous. The nodes communicate through the electric grid itself, so that transmission coverage is not an issue of importance.

This scenario clearly prevents an optimal educated guess, specially when only a small number of sensing nodes is available.

We define $\delta(s,x) = \omega_1(s,x) + \omega_2(s,x)$, where:

- $\omega_1(s,x)$ corresponds to propagation distance through wood space.
- $\omega_2(s,x)$ corresponds to propagation distance through open space.

Given a sensor location $s \in X$ and a point $x \in X$ we say that x is visible from sensor s if:

$$\frac{1 \text{ Km}}{750 \text{ m}}\omega_1(s,x) + \omega_2(s,x) < 1 \text{ Km} \tag{4.12}$$

Let $S = \{s_1 \ldots, s_p\}, s_k \in X, k = 1, \ldots, p$, be the positions of p acoustic sensors on X. We denote by $V(x)$ the set of all sensors $s \in S$ that are visible from the location x. We also define an arbitrary grid $G \subseteq X$. Objective function f_1, which measures acoustic sensor coverage, is defined as follows:

$$f_1(S) = \sum_{x \in G} \begin{cases} 0.5 \cdot \text{card}(V(x)) & \text{if card}(V(x)) < 3 \\ 3 + 0.01 \cdot (\text{card}(V(x)) - 3) & \text{if card}(V(x)) \geq 3 \end{cases} \tag{4.13}$$

This function penalizes grid points that see less than three sensors (a gunshot in those points cannot be located with highest precision), and gives a small bonus to grid points that see more than three sensors (the minimum number of sensors for highest precision location).

A second objective function f_2 measures the cost of the sensor deployment. As in the case of sensor coverage, there are many ways to model this. In this work we assume that the cost of a sensor unit is negligible compared to the cost of a permanent power line. As we previously said, the scenario is crossed by m power lines. We define:

$$f_2(S) = \sum_{s \in S} \min_{i \in \{1,\ldots,m\}} \delta_i(s) \tag{4.14}$$

where $\delta_i(s)$ is the Euclidean distance between s and the i-th power line, *i.e.*, between s and the point in that line that is closest to s in Euclidean distance.

Our ultimate task is to place p sensors on X in such a way that the coverage f_1 on X is maximized and the cost f_2 on X is minimized. Clearly, these objectives are contradictory. Minimizing $-f_1$ (*i.e.*, maximizing f_1) tends to spread the sensors, whereas minimizing f_2 tends to concentrate them around the power lines. For that reason, it is desirable to produce the **Pareto front** [GPBRGCn08] of these two functions, which represents a pool of *candidate* solutions. A point $x^* \in X$ belongs to the Pareto front of a set of functions in X if a further decrease of one of them is not possible without causing an increase in some of them. The methodology in [GPBRGCn08] obtains joint descent directions for all the objective functions in a set, but it requires all of them to be differentiable, and that is not the case of f_1.

In our case, since there are only two objective functions, we define the following unconstrained optimization problem:

$$\begin{aligned} &minimize \quad f(s_1, \ldots, s_p) = (\theta - 1) f_1(s_1, \ldots, s_p) + \theta f_2(s_1, \ldots, s_p) \\ &s_1, \ldots, s_p \end{aligned} \tag{4.15}$$

By solving problem (4.15) repeatedly, assigning random values to θ in $[0, 1]$, we obtain a collection of points of the Pareto front of f_1 and f_2.

Remark 1: Note that any local minima in $-f_1$ and f_2 belong, by definition, to the Pareto front.

Next section deals with the solution of the model, including a proper choice of its parameters.

4.3.4 Solving the optimization model

Many optimization algorithms are iterative. Starting with a solution estimate $S_1 = \{s_{11}, \ldots, s_{1p}\}$, a subsequence $\{S_i\}_{i\in I} = \{s_{i1}, \ldots, s_{ip}\}_{i\in I}$ is generated that hopefully converges to the solution of the problem. As it is common in all implementations, there are several parameters *(magic numbers)* the user must set. Some of them will notably influence the performance of the algorithm, and often depend upon the structure of the objective function.

4.3.4.1 Alternative approaches

Two methods were suggested in [KU02, UK03] for access point coverage optimization, a similar problem to ours: *neighborhood search* and *simulated annealing* [Loc00], which we compared in [GCnCMBRGP08]. These methods have no guaranteed convergence. On the other hand, gradient descent methods converge, but they can only we applied when the objective function is smooth, which is unusual in realistic models like ours.

4.3.4.2 Derivative-free unconstrained minimization

The function $f(S)$, with $\theta \in (0, 1]$, is non-smooth on X with directional derivatives everywhere defined, which is a required assumption on a recent algorithm for unconstrained minimization (ignoring $\theta = 0$ is not relevant to estimate the Pareto front, because θ can be arbitrarily close to 0). Numerical results show that the algorithm is competitive with others that try to find a good local minimum [GPR02, GPGCnBR06]. Essentially the algorithm is an iterative process that does not force the decrease of $f(S_i)$, but imposes a controlled bound $\varphi_i \geq f(S_i)$ at every iteration. More specifically, given a step size $\tau_i\Delta > 0$, and a unitary direction $D \in \mathbb{R}^{2p}, D = \{d_1, \ldots, d_p\}$, one iteration of the algorithm succeeds if

$$f(S_i + \tau_i\Delta D) \leq f(S_i) + \alpha_i(\varphi_i - f(S_i)) - \nu(\tau_i\Delta), \tag{4.16}$$

where $\nu(\cdot), \varphi$ satisfy **A4 - A5** given below. The point S_i is *blocked* when the algorithm fails to satisfy (4.16) on a set of directions $\{D_1, \ldots, D_n\}, n > 2p$ that positively spans $\mathbb{R}^{2p}$. It is shown in [GPGCnBR06] that under assumptions **A1 - A5** given below, the sequence of blocked points converges to a point S^* that satisfies the zero order stationary point of $f(S)$, *i.e.*, $f'(S^*, D_k) \geq 0, k = 1, \ldots, n$, where the directional derivative is nonnegative along the given directions. In theory, if **A6** also holds, then $\nabla f(S^*) = 0$, but we are aware that **A6** is seldom fulfilled for our kind of function and we do not stress this result. The reader may read [GPR02, GPGCnBR06] to complete the details. We reproduce [GPGCnBR06, table 1] in Table 4.12.

A1. $f(S) : \mathbb{R}^{2p} \to \mathbb{R}$ is bounded below, and $\{S_i\}_{i=1}^{\infty}$ remains in a compact set,

A2. $f(S)$ has directional derivatives $f'(S, D)$ everywhere defined:

$$\eta > 0 \Rightarrow \begin{cases} f'(S, \eta D) = \eta f'(S, D), \\ f(S + \eta D) = f(S) + \eta f'(S, D) + o(\eta) \end{cases} \tag{4.17}$$

A3. The unit directions $D_1, \ldots, D_n$ positively span R^{2p}.

A4. The function $\nu(\cdot) : \mathbb{R}_+ \to \mathbb{R}_+$ is *little-o* of τ, that is: $\lim_{\tau \downarrow 0} \nu(\tau)/\tau = 0$

A5. The reference values $\{\varphi_i\}_1^\infty$ are upper bounds of $f(\cdot)$, *i.e.*, $\varphi_i \geq f(S_i)$ for all i, and decrease sufficiently after a given finite number of successful iterations.

A6. $f(S)$ is strictly differentiable or locally convex at all limit points of the sequence $\{S_i\}_{i=1}^\infty$ generated by the algorithm.

Remark 2: The simplest case of INTERPOLATE$(S, D, \tau\Delta)$ is $Z = S + \tau\Delta D$, $f_Z = f(Z)$, although there exist more elaborate alternatives [GPGCnBR06].
Remark 3: If $\alpha = 0$, we obtain the monotone version of the derivative free algorithm, which converges to the local minimum in the neighborhood of the starting point.
Remark 4: **A5** holds.
Remark 5: In order to improve convergence, [GPGCnBR06] suggested to expand τ every time a *significant* number of successes is achieved. Nevertheless, convergence is also guaranteed if τ only decreases.

Regarding conditions **A1-A5**: (1) The objective function $f(\cdot) : \mathbb{R}^{2p} \to \mathbb{R}$ as defined in (4.15) is non negative by adding a constant and $S = \{s_1, \ldots, s_p\}$ remains in the compact set X, (2) $f(\cdot)$ possesses everywhere directional derivatives if $\theta \in (0, 1]$, (3) the directions of search $D_k \in \mathbb{R}^{2p}, k = 1, \ldots, n$ are easy to generate [GPR02], (4) the choice of $\nu(\tau\Delta)$ in Section 4.3.5 is $0.001(\tau\Delta)^2$, and (5) $\varphi = f(S)$ as soon as all directions in D have been explored (Table 4.12 and remark 4).

4.3.5 Example of deployment

Deployment target is a 10 km $\times$ 10 km flat wood scenario shown in Figure 4.13. The light areas represent open space, and the dark ones represent wood patches. The scenario is crossed by three "vertical" power lines, respectively at 1.66, 5 and 8.33 Km from the left side. This terrain representation partly encloses the difficulties in modeling Cabañeros national park.

Grid G is given by a uniform discretization of the scenario in 50-m steps. In it we want to install 200 acoustic sensors ($p = 200$) with maximum detection coverage, at a minimum power cost. In order to obtain the Pareto front, we obtain the solution of (4.15) several times, starting from different random points S_0. We normalize the objective function by dividing it by its value at the starting points.

Since an algorithm can lead to a local optimum of (4.11) or to a solution that differs from the global optimum, we have chosen a brute force approach that computes the function in every grid point. Real-time system response is not affected, as the calculation time in a 10 km $\times$ 10 km scenario is in the order of a tenth of millisecond.

Following the results in [GCnCMBRGP08], instead of applying the non-monotone derivative-free search as described in Section 4.3.4, we applied a *zone search* variant. If we simply "move" one sensor at a time, instead of "moving" them all, the evaluation of the objective function is significantly less time consuming, since most computations in $f(S)$ do not change.

Parameters: $\epsilon_\tau, \mu, \tau, \varphi$

Get S, let $f_S = f(S)$

DO `success`$= 0$

Choose D_k, $k = 1, \ldots, n$ that positively span $\mathbb{R}^{2p}$

FOR $j = 1$ TO n

$[Z\ f_Z] =$ INTERPOLATE $(S, D_k, \tau\Delta)$ Remark 2

$\alpha = \min(\tau, \alpha), \phi = f_S + \alpha(\varphi - f_S)$ Remark 3

IF $(f_Z \leq \phi - \nu(\tau\Delta))$

`success=success+1`

$S = Z, f_S = f_Z$

ENDIF

ENDFOR

$\varphi = f_S$ Remark 4

IF (`success>max_success`) Remark 5

$\tau = \tau + 1$

ELSE

$\tau = \tau - 1$

ENDIF

WHILE $(\tau > \epsilon_\tau)$

Figure 4.12: Non-monotone derivative-free algorithm.

To formalize this approach, we split the scenario into q non-overlapping zones X_j, $j = 1, \ldots, q$, such that $X = \cup_{j=1..q} X_j$ and $X_k \cap X_j = \emptyset, k \neq j$. Let $V(s_i) = \{x \in G \mid s_i \in V(x)\}$. When we move the k-th sensor, the remaining sensor positions do not change, *i.e.*, the group moves from $S = \{s_1, \ldots, s_k, \ldots, s_p\}$ to $S' = \{s_1, \ldots, s'_k, \ldots, s_p\}$. When this happens, $V(S') = V(S) \cup V(s'_k) - V(s_k)$. To quickly obtain a *relaxed* estimate T_v of $V(s_k) \cup V(s'_k)$, we discard all $X_j \mid \forall x \in X_j \ \min(\delta(s_k, x), \delta(s'_k, x)) > 1$ Km. Let T_{nv} be the set of discarded zones and let $T_v = V(S') - T_{nv}$. Then, we compute $\sum_{x \in T_v} f(S')$, and we keep $\sum_{x \in T_{nv}} f(S') = \sum_{x \in T_{nv}} f(S)$ from the previous iterate. In the numerical tests that follow, we divided our scenario in 100 [1 Km $\times$ 1 Km] zones ($q = 100$). We compute one objective function component per zone at the beginning of the algorithm execution. When only one sensor is moved, there is no need to recompute the objective function in zones that are not affected by the sensor movement.

This way, the time to compute an objective function value drops from 20 seconds in average to just 1.5 seconds on a Pentium IV.

Let $S = \{s_1, \ldots, s_k, \ldots, s_p\}$. In order to move only one AP at a time, say s_k, we

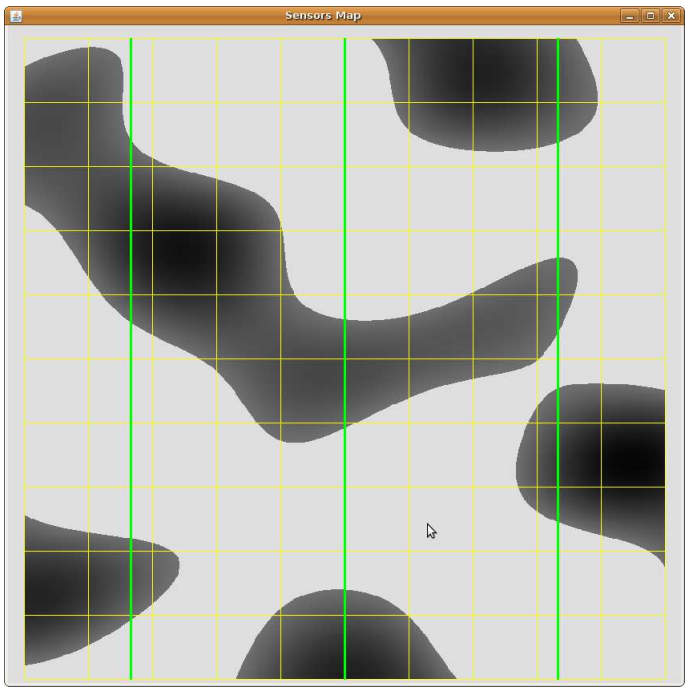


Figure 4.13: Synthetic outdoor scenario for the numerical tests. The green lines represent power lines.

generate a set of unit search directions $d_j \in \mathbb{R}^2, j = 1, \ldots, n$ such that the set $D = \{d_1, \ldots, d_n\}$ positively spans $\mathbb{R}^2$. We recall that $n \geq 3$. We declare a *success* when

$$f(s_1, \ldots, s_{k-1}, s_k + \tau_i d_j, s_{k+1}, \ldots, s_p) \leq f(S) + \alpha_i(\varphi_i - f(S)) - \nu(\tau_i)$$

for some $d_j \in D$ and some $s \in S$. The point S is *blocked* if the algorithm is unable to move a single $s \in S$. We observe that this schema may be carried out simultaneously on a multi processor environment and it is straightforward to show that after we try all $s \in S$ we have searched on a set of directions that positively span $\mathbb{R}^{2p}$, although we are using at least $3p$ directions of search.

We tuned the method in preliminary trials and determined the following parameter values:

- $\Delta = 0.1$ Km, *i.e.*, the method stops when the maximum sensor displacement is under 100 m.
- $\epsilon_\tau = 1$
- $\tau = 5$
- `max_success`$= 40$ (20%p), but we do not allow $\tau > 8$.
- We initially set φ to the value of the objective function at the starting point.
- $\alpha = 0$, *i.e.*, we perform a *monotone* derivative-free search.

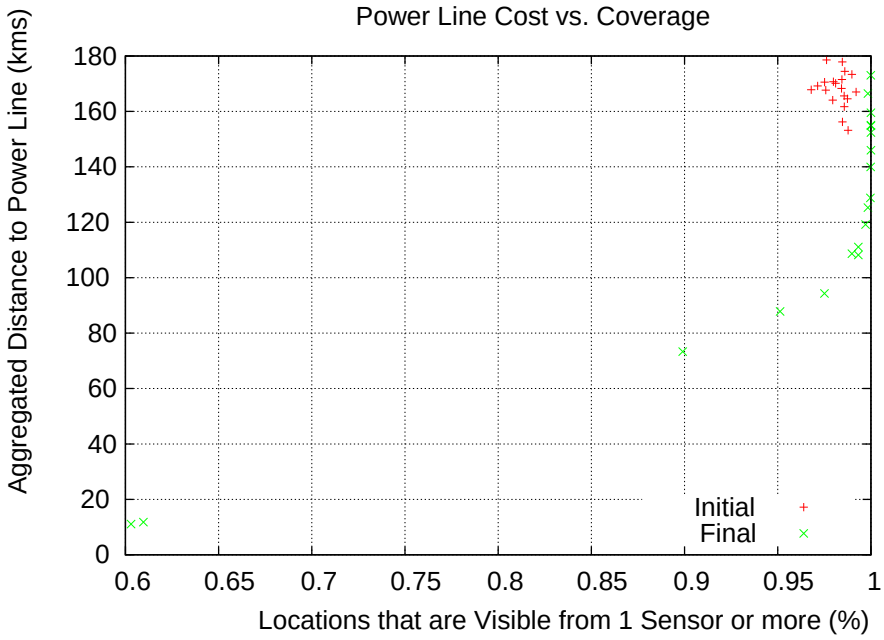


Figure 4.14: Aggregated distance to the power lines (cost) vs. coverage area: $V(x) \geq 1$.

- Finally, we set $\nu(\tau\Delta) = 0.001(\tau\Delta)^2$.

Figures 4.14, 4.15 and 4.16 show the results. Instead of representing f_2 versus f_1, we represent f_2 versus the coverage areas (in percentage) that correspond to $V(x) \geq 1$, $V(x) \geq 2$ and $V(x) \geq 3$, respectively. Algorithm execution is repeated 20 times, from 20 different starting points. Each starting point consists of a random deployment of 200 acoustic sensors in the scenario in Figure 4.13, with a uniform distribution.

The results reveal that the Pareto front approach is useful. The solutions show a compromise between cost and coverage. In the $V(x) \geq 1$ case there is no real advantage over a random seeding in terms of coverage, although the cost drops considerably for 100% coverage. However, in the $V(x) \geq 3$ case, the optimization result is clearly superior to random seeding across the whole Pareto front. At maximum cost, there is a 20% increase in coverage, and at 85% coverage (the best coverage of random seeding) there is a 50% decrease in cost.

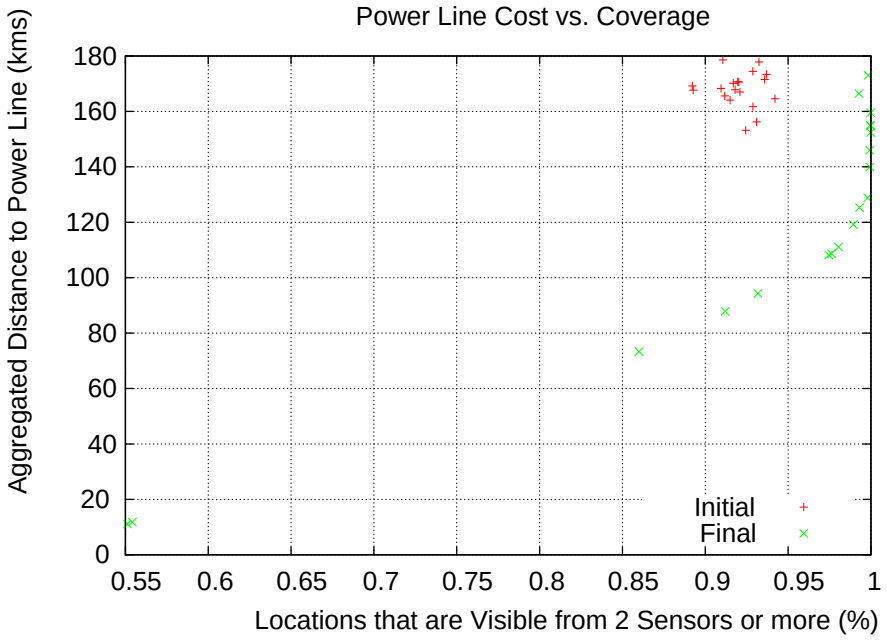


Figure 4.15: Aggregated distance to the power lines (cost) vs. coverage area: $V(x) \geq 2$.

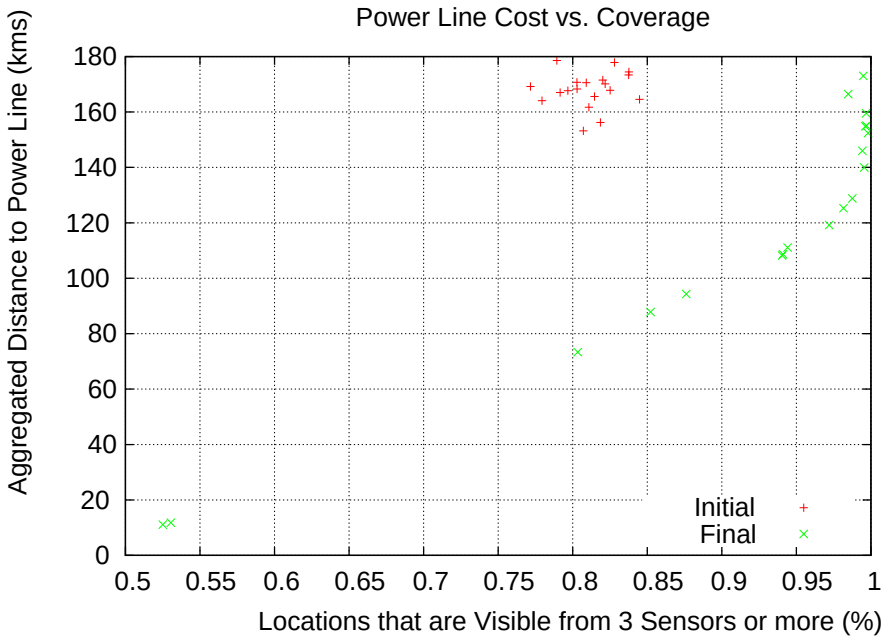


Figure 4.16: Aggregated distance to the power lines (cost) vs. coverage area: $V(x) \geq 3$.

Figures 4.17, 4.18 and 4.19 show three sensor deployments in our synthetic scenario in Figure 4.13, for three different choices of $\theta = \{0.1, 0.5, 0.9\}$ (red points represent sensor positions, blue areas zones with less coverage, dark blue areas indicate no coverage at

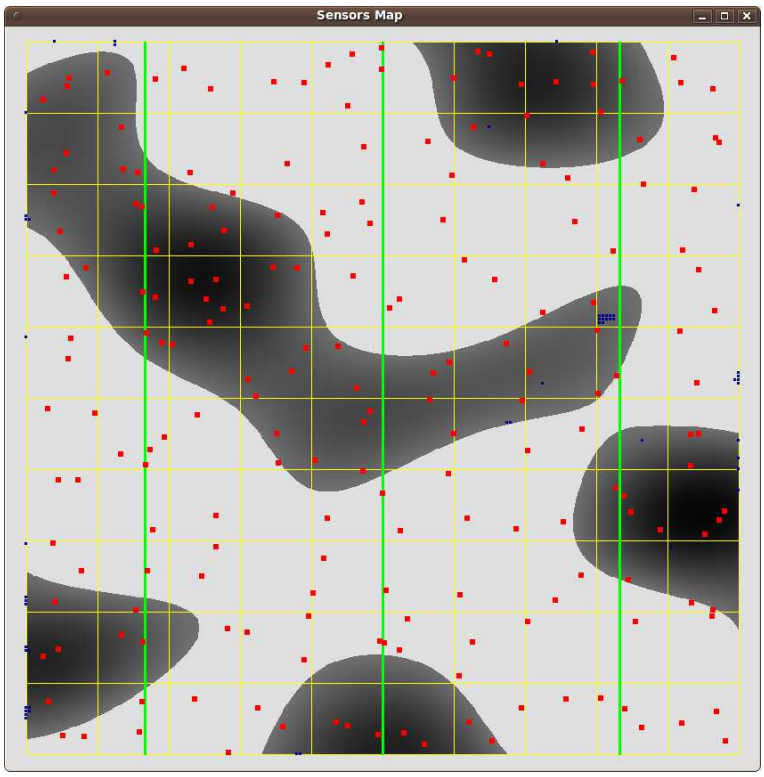


Figure 4.17: A sensor deployment for $\theta = 0.1$.

all). The results are consistent with the fact that coverage improves for low θ values. As the values of θ become higher, the sensors tend to concentrate around the power lines. Therefore, power cost decreases, but so the coverage does (Figure 4.19 for $\theta = 0.9$).

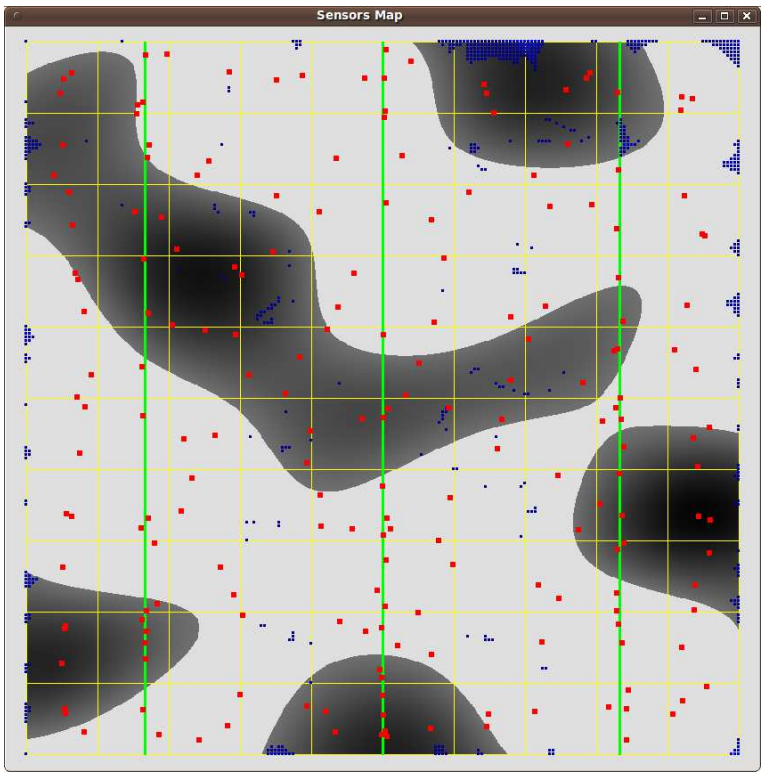


Figure 4.18: A sensor deployment for $\theta = 0.5$.

Figure 4.20 represents the percentage of grid points covered by different numbers of sensors in the three previous deployments. In the $\theta = 0.9$ case, where the cost function has more weight, over 35% grid points are out of coverage, and, due to the concentration of the sensors around the power lines, there are grid points that are covered by 13 sensors or more. As θ decreases, grid points covered by less than 3 sensors are rare, and the majority of the grid points are covered by 3-5 sensors.

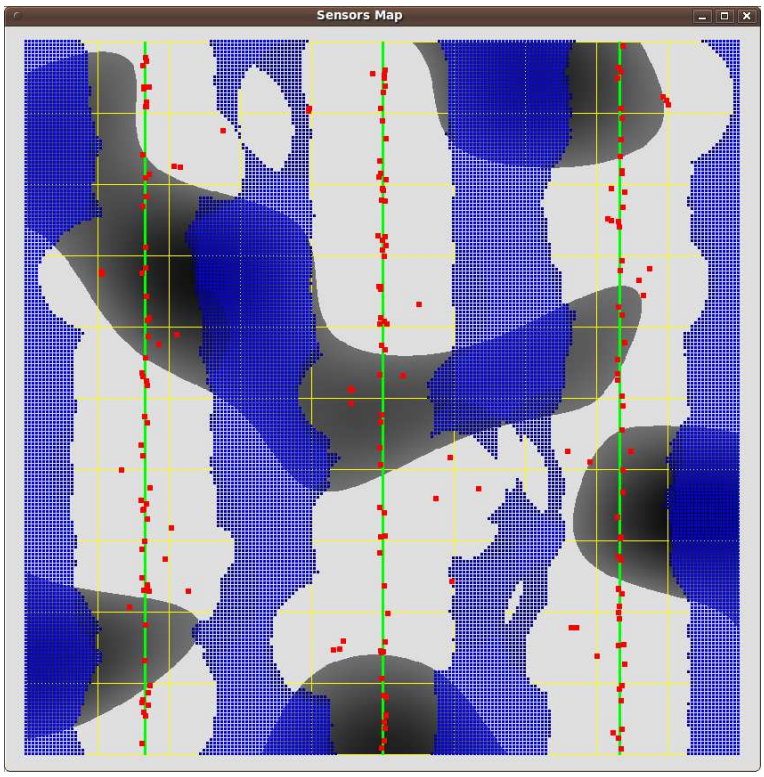


Figure 4.19: A sensor deployment for $\theta = 0.9$.

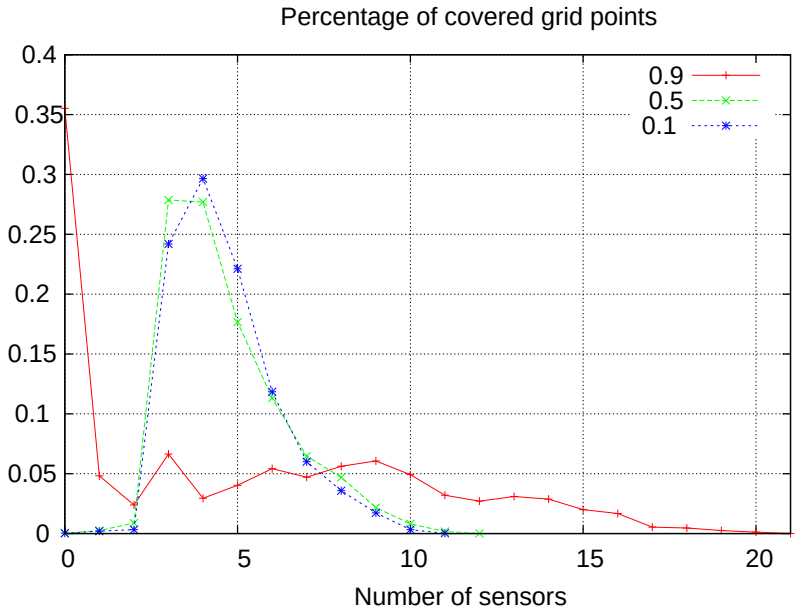


Figure 4.20: Percentage of covered grid points versus number of detecting sensors for different values of θ.

		>1	>2	>3	>4	>5
		[0.99973%]	[0.99777%]	[0.99448%]	[0.75258%]	[0.45608%]
$\theta = 0.1$	sync. $< 1ms$	64.40	64.40	64.40	5.30	1.60
	sync. 1ms	78.00	66.90	73.90	5.50	0.00
	sync. 10ms	76.90	90.70	73.40	13.60	3.90
		>1	>2	>3	>4	>5
		[0.99988%]	[0.99730%]	[0.98856%]	[0.70991%]	[0.43301%]
$\theta = 0.5$	sync. $< 1ms$	139.20	138.90	123.50	8.30	0.00
	sync. 1ms	146.70	120.20	125.40	22.50	0.00
	sync. 10ms	144.90	162.60	150.20	42.70	4.00
		>1	>2	>3	>4	>5
		[0.64476%]	[0.59659%]	[0.57273%]	[0.50632%]	[0.47687%]
$\theta = 0.9$	sync. $< 1ms$	114.00	66.80	20.20	11.00	1.50
	sync. 1ms	99.30	56.40	49.90	19.20	2.40
	sync. 10ms	171.80	127.10	88.00	31.70	19.50

Table 4.1: Gunshot average location error (m).

Table 4.1 shows location precision and expected coverage results. The location algorithm described in Section 4.3.2.2 has been tested in the three previous deployments ($\theta = \{0.1, 0.5, 0.9\}$). Table 4.1 shows the average location error in meters (distance between estimated and real positions) for $10,000$ random gunshot positions in the simulation area, for each θ value and three levels of variation in sensor clocks. These variations are generated by adding a normal Gaussian random toss to the arrival time at each node, for three different values of σ between 0 and 10 ms. The results show that location accuracy increases with the number of detecting nodes, as well as with a better clock synchronization. In addition, this table provides information on the expected coverage at each scenario. For instance, in the $\theta = 0.1$ case, the probability that one or more nodes detect the event is 0.99973, for two or more nodes it is 0.99777, and so on. Clearly, there must be a compromise between detection accuracy, coverage and deployment cost.

4.4 Summary

In this chapter we solved two deterministic deployment problems. Our optimization models are able to plan the position of the nodes while accomplishing optimization goals.

First, we developed a methodology based on the $ACO_{\mathbb{R}}$ metaheuristic for sensor node placement. The algorithm was tested in two distinct scenarios (convex and non-convex) against the FDP heuristic. $ACO_{\mathbb{R}}$ outperformed FDP in all the tests in both cases. The time consumed by the algorithm also scales well with the size of the search space (number of nodes and map size). Moreover, the methodology developed can be easily extended to

other complex placement problems in almost any AmI outdoor scenarios.

Second, we proposed a system that detects and locates gunshots by means of a WSN which detects acoustic signals. In this case a Pareto frontier approach is selected to characterize solutions of large scale deployments, where a trade-off must be established between cost and detection accuracy. The network planning optimization is based on a monotone descent method without derivatives that is compatible with realistic optimization functions. The location procedure is based on hyperbolic multilateration using data from time-synchronized sensor network. We also propose a practical distributed synchronization algorithm for that purpose, with low energy consumption. Our results are clearly superior to unguided placement, achieving a 50% cost reduction for 85%coverage. In the two-dimensional scenario, we show that event detection by at least four nodes is required to reach a satisfactory gunshot location accuracy.

Chapter 5

Random deployment of AmI networks

In this chapter we address an optimization random deployment. As in the previous chapter we consider the realistic assumption of non-homogeneous importance scenarios. Our goal is to maximize the importance captured subject to a *partially-controlled* node placement. These connectivity constraints are related to the information conveyed to the sinks, as in deterministic deployments. However, in this case we can not guarantee a fully connected network, due to the uncertainty in individual node position. Rather, we express connectivity as a probabilistic model.

Moreover, in our model we assume that nodes are grouped in several clusters, which are spread following Gaussian random distributions. The goal is to decide the launch point and dispersion for each cluster. This corresponds to real situations where clusters are dropped in an airborne launch controlling the dispersion with the releasing altitude. The problem is solved by dividing the optimization problem in steps where single cluster deployments are addressed.

The generality of our model allows us to handle target areas of any shape and distribution of importance. In particular, for the examples provided in this chapter we selected the lunar Tycho crater as the target scenario (see Figure 5.4) and for our validation tests.

5.1 Introduction and scope

We consider a special *integrated coverage and connectivity* problem [GD08] is addressed for random deployments. The goal is to find the optimal network deployment scheme that maximizes the joint sensing coverage of the nodes, while satisfying connectivity conditions. Our system model considers the following major features:

1. Node placement is inherently random, due to airborne sensor deployment, for example. Nevertheless, it is assumed that the network designer can select certain placement parameters, which allows some control over the position of the nodes.

2. The nodes are grouped in one or more *clusters*, and each cluster placement can be independently controlled. In this work, Gaussian random distributions are assumed for the clusters, since they correspond to actual observed distributions in airborne seeding [For04, ZC03, FDN05]. For each cluster two parameters can be selected:

the *target point* (*i.e.* where the nodes are aimed) and the *dispersion* (*i.e.* the standard deviation). The latter controls how the nodes are spread from such points and may be easily controlled in airborne placements since it is related to the altitude at which the cluster is released (see [For04]).

3. Some zones in the target area are more "important" than others (*non-homogeneous* scenario), distinguishing our model from many previous models which assume a uniformly distributed importance across the target area (see Section 1.3.3.2). In our model a map describes the importance of each point in the operational region. It is assumed that sensors collect information from a small area (sensing coverage), and that overlapping does not provide additional information. Therefore nodes must be spread to avoid overlapping of sensing areas.

4. Sensed data must be delivered to a control node (*sink* node in WSN terminology), otherwise data will be lost. The routing capabilities of WSN make multi-hop transmissions possible, so sensed data can be delivered whenever a path is available. However, some nodes may be isolated (due to random scattering), and the information they capture cannot reach the sink.

Therefore, the challenge is to select suitable target points and dispersions which provide a trade-off between sensing overlapping and the number of isolated nodes. This leads to a complex stochastic optimization problem. As will be discussed later in Section 5.4, this problem can be efficiently addressed by dividing the optimization into several steps (one per cluster), where simplified placement problems are addressed separately.

Let us remark that, although energy constraints are important in the design of a WSN, we focus exclusively on sensing coverage; other constraints are beyond the scope of this work.

Without loss of generality, this model is inspired by planetary exploration scenarios such as those proposed in [GN06, DIL+05], where the operational region is notably larger than the sensing range of a node. So far, inter–planetary exploration probes and rovers have only been able to land in and explore small areas in a few solar system bodies such as Mars or Saturn's moon Titan. In this scenario the "important areas" may be, for instance, places where the best chances of finding traces of water exist, as has been the goal of NASA's Mars Sojourn rovers Spirit and Opportunity [SPI11].

Within this context, WSN may become a feasible alternative for extending monitoring range. Clusters can be deployed over large important areas, and simple nodes would be in charge of data monitoring. Besides, these nodes, rather than deliver the data directly to Earth (which would require complex communication hardware) would relay it via the sink (a special node with such capability). In addition, the Gaussian distribution, selected to model the network placement, also fits with this scenario. During the terminal descent phase of the probe, the expulsion mechanism can be adjusted to release the nodes at a specific altitude above the surface [DIL+05].

5.2 Major contributions

In this Chapter we analytically solve the optimality of a WSN random deployment considering Gaussian clusters. Precedent researches have mainly focused on network connectivity issues in non-clustered WSNs. Moreover, we also integrate the effect of non-homogeneous importance in the target scenario, unlike previous works which have largely ignored it. Let us remark that our analysis can be also applied to homogeneous scenarios (which is a particular case of the non-homogeneous problem). Section 1.3.3.2 discusses these related works.

In summary, the main contributions of this Chapter are:

- We obtain a mathematical optimization problem from the system model (Section 5.3). This problem fully integrates both coverage and connectivity issues for clustered networks.

- We demonstrate how the global problem naturally decomposes into smaller problems, each related to the deployment of a single cluster. These problems can be solved simultaneously, leading to a *global optimization* strategy. As will be discussed later, analytical solutions in this field are still unknown. In this Chapter we develop an approximation to characterize the joint sensing coverage of a cluster. Several tests have been performed on this approach and the results (Section 5.5) clearly validate it.

- We show that an approximate *iterative optimization* achieves solutions close to the global optimization solutions in complex scenarios while being computationally feasible for any number of clusters, unlike the global optimization which does not scale up well.

- Finally, we have tested this methodology in complex scenarios, including the lunar Tycho crater as the target scenario (see Figure 5.4) for our validation tests. Providing coverage to such an extension is a challenging project as will be shown in Section 5.5.

5.3 Sensor placement model

In this section we characterize the deployment scenario and the model of our network, and describe the underlying assumptions which define the constrained optimization problem.

Our model comprises M independent *clusters* of nodes, each composed by N sensors (including a sink node). These clusters must be deployed in a non-homogeneous target area, such that the importance (quality) of the information captured is maximized. The planning is subject to communication constraints, since nodes must have a path (either direct or via a multi-hop route) to their corresponding sink node.

Figure 5.1 illustrates the main concepts of the model. The deployment area is a euclidean space $\mathcal{X} \subseteq \mathbb{R}^2$ (with distance function $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$). Besides, there exists a

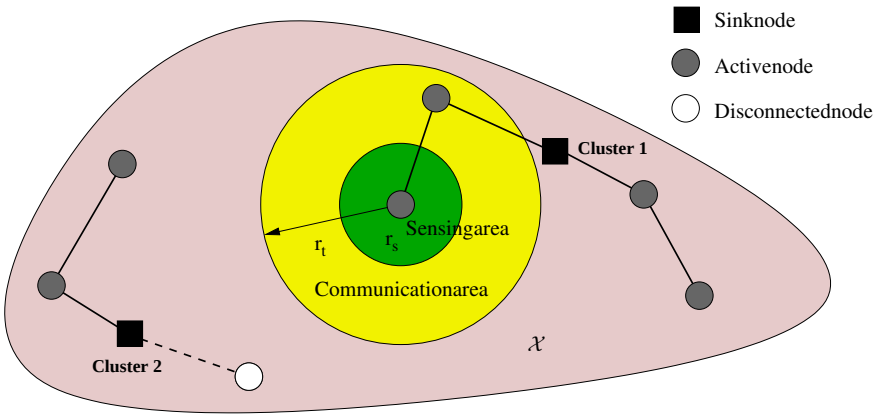


Figure 5.1: System model.

value of importance associated with every $x \in \mathcal{X}$ which is defined by a real and continuous *importance function* $\alpha : \mathcal{X} \to \mathbb{R}_0^+$. Therefore, the total importance contained in the target area can be computed as:

$$\Gamma_{max} = \int_{\mathcal{X}} \alpha(x) \mathrm{d}x \tag{5.1}$$

Let $\Phi_j = \{\varphi_i^j : i = 1, \ldots, N\}$ denote the nodes in cluster j for $j = 1, \ldots, M$, and $\mathcal{S}_j = \{x_i^j : i = 1, \ldots, N\}$ are the sets which contain the positions of the nodes in the target area.

For the sensor network architecture the following considerations have been established:

- The sensing hardware is homogeneous, *i.e.* sensor nodes are of the same type and have the same sensing capabilities.
- Each cluster operates independently, *i.e.* nodes of different clusters do not communicate.
- The network sink can be any of the N sensor nodes. Without loss of generality, we assume that nodes φ_1^j for $j = 1, \ldots, M$ are the sinks and x_1^j for $j = 1, \ldots, M$ their positions.
- The *transmission range* r_t is the longest distance between two mutually communicating nodes.
- The *sensing range* r_s is the longest distance between the node and the locations which it is able to sense. It is assumed that a sensor is able to capture all the importance within its sensing range. Therefore the importance covered by sensor φ_i^j is:

$$\int_{B(x_i^j, r_s)} \alpha(x) \mathrm{d}x \tag{5.2}$$

being $B(x, r)$ the open ball in $\mathbb{R}^2$ centered in x with radius r.

- The set $\Phi_j^\star \subseteq \Phi_j$ for $j = 1, \ldots, M$, contains the connected cluster of nodes able to transmit sensed data to their corresponding sink φ_1^j, either directly or by multi-hop transmissions. Set $\Phi_j^\star$ includes the sink itself. Hereafter, let us refer to the nodes belonging to $\Phi_j^\star$ as *active* nodes.

In our model it is assumed that, for each cluster j, nodes are spread randomly around the target point of this cluster (thereby denoted as x_*^j), following independent Gaussian distributions $N(0, \sigma_j)$ in each axis. Let us denote Δ_j as the bivariate Gaussian distribution which measures this displacement. Then, the position of the nodes in the cluster j is given by independent identically distributed random variables $x_i^j = x_*^j + \Delta_j$ for $i = 1, \ldots, N$.

We selected Gaussian distributions for some of their particular properties. First, it has been demonstrated (see [JB03, GBGL08]) that in the experiment of throwing an object aimed at a particular location, the error can be modeled as independent Gaussian variables in each axis. The resulting density function depends on the distance from the target point, but not on its coordinates (rotationally invariant property). This property can be applied to predict the position of every sensor node in a cluster, aimed at the same target point, like in our model. Moreover, Gaussian dispersion can be modulated in an airborne deployment, since the final position error is correlated with the cluster releasing altitude (see [For04]). Thus, the expulsion mechanism can be adjusted to release the nodes at an specific altitude above the surface, providing certain control over the final position of the nodes.

Finally, let us assume that the importance captured from an open region $\mathcal{A} \subseteq \mathcal{X}$ is maximal if for all $a \in \mathcal{A}$ there exists at least one active node $\varphi_i^j \in \Phi_j^\star$ for some $j \in [1, M]$ such that $a \in B(x_i^j, r_s)$. In other words, if the sensing areas of active nodes overlap, the importance captured in those areas does not increase. Therefore, the open set describing the sensed area is:

$$\mathcal{B} = \left(\bigcup_{j=1}^{M} \bigcup_{i \in \Phi_j^\star} B(x_i^j, r_s) \right) \bigcap \mathcal{X} \tag{5.3}$$

Let us remark that since node position is random, $\mathcal{B}$ is random as well. Therefore the importance captured in a network deployment is also random. The next section proposes a stochastic mathematical program for our scenario.

5.4 Optimization model

The question addressed in this section is how to spread the sensors over the designated areas to maximize the expected covered importance.

Let $\mathbf{1}_\mathcal{B}(x)$ denote the indicator function equal to 1 if $x \in \mathcal{B}$ and 0 otherwise. The optimization problem describing our model can be stated as:

Given

$$\alpha : \mathcal{X} \to \mathbb{R}_0^+$$

find

$$x_*^j, \sigma_j, \text{ for } j = 1, \ldots, M$$

such that

$$\max_{\substack{x_*^j, \sigma_j \\ j=1,\ldots,M}} \mathbb{E}\left[\int_{\mathcal{X}} \alpha(x)\mathbf{1}_{\mathcal{B}}(x)\mathrm{d}x\right] \tag{5.4}$$

subject to

$$x_*^j \in \mathcal{X}$$
$$\sigma_j > 0$$

for $j = 1, \ldots, M$.

Let us consider an example area with non-homogeneous importance, like the one selected for the sample scenario shown in Figure 5.4. Clearly, in large/complex scenarios manual placement is not possible. Besides, direct algorithmic computation of the previous optimization problem is unfeasible: note that (5.4) requires the calculation of $\mathbb{E}\left[\mathbf{1}_{\mathcal{B}}(x)\right] = \Pr\left[x \in \mathcal{B}\right]$ given any particular combination of network configuration parameters x_*^j, σ_j, for j=1, $\ldots, M$. If all the nodes in a cluster belong to its active set, this probability can be easily derived (following the method used later in Section 5.4.1). However, this is an oversimplification since $\mathcal{B}$ is the area covered *only by active nodes*. The actual probability computation leads to a connectivity problem which falls in the field of percolation theory in RGGs, but general results for Gaussian clusters are still unknown [PP03]. Therefore, in this work, we propose an approximated computation of (5.4) using a mixed analytical-numerical approach which is described in the following sections.

In order to find a suitable optimization model let us recall that clusters operate independently. This suggests that the whole expected importance can be computed as the sum of the expected importances captured by each separate cluster. In fact, (5.4) can be rewritten as:

$$\max_{\substack{x_*^j, \sigma_j \\ j=1,\ldots,M}} \mathbb{E}\left[\int_{\mathcal{X}} \alpha(x)\mathbf{1}_{\bigcup_{j=1}^{M} \mathcal{B}^j}(x)\mathrm{d}x\right] \tag{5.5}$$

where $\mathcal{B}^j$ is the area covered by the cluster j, that is,

$$\mathcal{B}^j = \bigcup_{\varphi_i^j \in \Phi_j^\star} B(x_i^j, r_s)$$

Let $\mathbf{0}_{\mathcal{B}}(x)$ denote the inverse indicator function equal to 0 if $x \in \mathcal{B}$ and 1 otherwise. Reorganizing formula (5.5):

$$\max_{\substack{x_*^j,\sigma_j \\ j=1,\dots,M}} \mathbb{E}\Bigg[\int_{\mathcal{X}} \alpha(x)\mathbf{1}_{\mathcal{B}^1}(x)\mathrm{d}x + \int_{\mathcal{X}} \alpha(x)\mathbf{0}_{\mathcal{B}^1}(x)\mathbf{1}_{\mathcal{B}^2}(x)\mathrm{d}x + \dots \\ \dots + \int_{\mathcal{X}} \alpha(x)\prod_{j=1}^{M-1}\mathbf{0}_{\mathcal{B}^j}(x)\mathbf{1}_{\mathcal{B}^M}(x)\mathrm{d}x\Bigg] \tag{5.6}$$

In words, for iteration j the area already covered by previous clusters must be subtracted (multiplying $\alpha(x)$ by $\prod_{j=1}^{j-1}\mathbf{0}_{\mathcal{B}^j}(x)$). The previous equation can be written more compactly by denoting:

$$\alpha^j(x) = \alpha^{j-1}(x)\mathbf{0}_{\mathcal{B}^{j-1}}(x), \text{ if } j > 1$$
$$\alpha^1(x) = \alpha(x)$$

Thus, $\alpha^j(x)$ represents the *remaining* importance at iteration j. Using this notation and since the expectation is a linear operator, the problem is to:

$$\max_{\substack{x_*^j,\sigma_j \\ j=1,\dots,M}} \sum_{j=1}^{M}\mathbb{E}\left[\int_{\mathcal{X}} \alpha^j(x)\mathbf{1}_{\mathcal{B}^j}(x)\mathrm{d}x\right] \tag{5.7}$$

By applying Fubini's theorem the order of integration and expectation (note that in a continuous domain this is actually a finite integral) can be changed, and leads to:

$$\max_{\substack{x_*^j,\sigma_j \\ j=1,\dots,M}} \sum_{j=1}^{M}\int_{\mathcal{X}} \mathbb{E}\left[\alpha^j(x)\mathbf{1}_{\mathcal{B}^j}(x)\right]\mathrm{d}x \tag{5.8}$$

The expectation inside the integral of the last formula can be decomposed as:

$$\mathbb{E}\left[\alpha^j(x)\mathbf{1}_{\mathcal{B}^j}(x)\right] = \mathbb{E}\left[\alpha(x)\prod_{k=1}^{j-1}\mathbf{0}_{\mathcal{B}^k}(x)\mathbf{1}_{\mathcal{B}^j}(x)\right] \tag{5.9}$$

Note that for any $j = 1,\dots,M$, the random sets $\mathcal{B}^k$, for $k = 1,\dots,j$ are independently determined. Thus, the random variables $0_{\mathcal{B}^k}(x)$, for $k = 1,\dots,j-1$ are also mutually independent and independent of $1_{\mathcal{B}^j}(x)$. Hence,

$$\mathbb{E}\left[\alpha^j(x)\mathbf{1}_{\mathcal{B}^j}(x)\right] = \alpha(x)\prod_{k=1}^{j-1}\mathbb{E}\left[\mathbf{1}_{\mathcal{B}^j}(x)\mathbb{E}\left[\mathbf{0}_{\mathcal{B}^k}(x)\right]\right] \tag{5.10}$$

Besides,

$$\mathbb{E}\left[\mathbf{1}_{\mathcal{B}^j}(x)\right] = \Pr\left[x \in \mathcal{B}^j\right] = 1 - \mathbb{E}\left[\mathbf{0}_{\mathcal{B}^j}(x)\right]$$

Finally, the problem is expressed as:

$$\max_{\substack{x_*^j,\sigma_j \\ j=1,\ldots,M}} \sum_{j=1}^{M} \int_{\mathcal{X}} \mathbb{E}\left[\alpha^j(x)\right] \Pr\left[x \in \mathcal{B}^j\right] \mathrm{d}x \tag{5.11}$$

where,

$$\mathbb{E}\left[\alpha^j(x)\right] = \alpha(x) \prod_{k=1}^{j-1} \Pr\left[x \notin \mathcal{B}^k\right]$$

Thus, we have proved that the problem is reduced to smaller problems each related to a single cluster. The solution to these individual problems is analyzed in the next section.

5.4.1 Single cluster analysis

This section addresses the problem of determining the probability that a point $x \in \mathcal{X}$ is covered by active nodes of a given cluster j, *i.e.* computing $\Pr\left[x \in \mathcal{B}^j\right]$. To simplify the notation, the script j referring to cluster j is omitted through this section. The strategy to obtain this probability is to approximate the importance captured by a cluster (let Γ denote such importance) and to then relate it to the analytical importance:

$$\int_{\mathcal{X}} \alpha(x) \Pr\left[x \in \mathcal{B}\right] \mathrm{d}x \tag{5.12}$$

To start the computation of Γ let us recall from Section 5.3 that the position of each sensor in a cluster is an independent identically distributed bivariate random variable $x_* + \Delta$ whose joint probability density function is given by:

$$f_{x_*+\Delta}(x; x_*, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{d(x, x_*)^2}{2\sigma^2}\right).$$

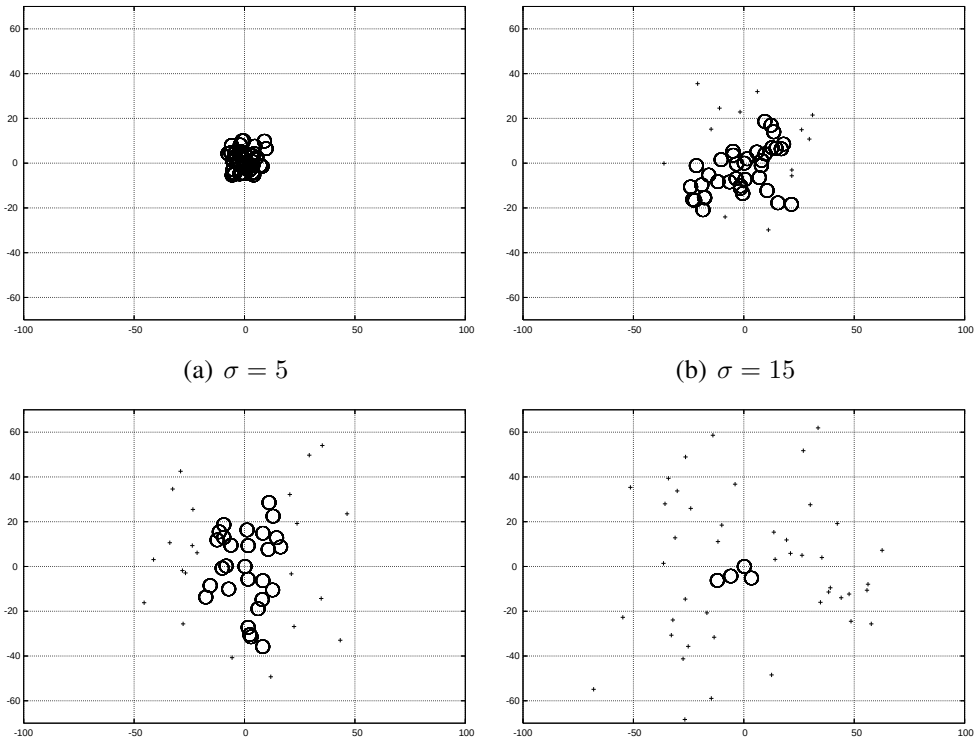
A node placed at position x with r_s sensing range will then capture the importance

$$\int_{\mathcal{B}(x,r_s)} \alpha(x)\mathrm{d}x \approx \pi r_s^2 \alpha(x) \tag{5.13}$$

Approximation holds if importance is uniform for scales comparable to r_s. Since the sensing ranges are usually small and the terrain does not change abruptly (note that $\alpha(x)$ is assumed to be continuous) approximation in (5.13) is taken in this work. Therefore, the *expected* importance I captured by a single sensor is:

$$I \approx \int_{\mathcal{X}} \pi r_s^2 \alpha(x) f_{x_*+\Delta}(x; x_*, \sigma)\mathrm{d}x \tag{5.14}$$

At first glance the expected importance captured by the N nodes in the cluster could be estimated as $N \times I$. This approach is equivalent to assuming that all the information can be delivered to the sink and that the sensing areas of all the nodes are non-overlapping. However, this is an oversimplification as discussed before. Figure 5.2 shows some random deployments aimed at position $(0, 0)$ where σ has values 5, 15, 25, and 35, $r_s = 3$ and $r_t = 10$. Active nodes are depicted with their sensing areas (circles) and disconnected



(a) $\sigma = 5$ (b) $\sigma = 15$

(c) $\sigma = 25$ (d) $\sigma = 35$

Figure 5.2: Effect of sensor dispersion in the importance captured. Importance is captured only in the circled areas.

nodes are marked with '+' symbols. Clearly, for low σ values most nodes are connected but sensing areas overlap, and thus deployment is inefficient. On the other hand, for high values of σ the deployment is also inefficient because of the sub-connected topology. In our example, the best configurations are achieved for mid dispersions.

Therefore, the goal is to select a balanced σ that keeps connectivity as high as possible while avoiding overlapping of sensing areas. To characterize this problem, let Ω denote the average number of active nodes in the cluster. In addition, let θ denote the ratio of overlapped sensing areas. Then, the captured importance of the cluster, Γ, can be computed as:

$$\Gamma \approx I(\Omega(1-\theta)+\theta) \tag{5.15}$$

Note that this is an approximation. If $\theta = 0$ (no overlapping), then the Ω nodes capture on average $I \times \Omega$. On the other hand, if the covered areas of all the nodes overlap, then $\theta = 1$, *i.e.* the captured importance corresponds to a single node (I). Moreover, note that Ω and θ depend on σ but are independent of the cluster's central position x_*. However, I depends on both σ and x_*.

Therefore, we obtain:

$$I(\Omega(1-\theta)+\theta) \approx \int_{\mathcal{X}} \alpha(x) \Pr\left[x \in \mathcal{B}\right] \mathrm{d}x \tag{5.16}$$

Substituting I by the approximation of (5.14):

$$\int_{\mathcal{X}} \pi r_s^2 \alpha(x) f_{x_*+\Delta}(x; x_*, \sigma)(\Omega(1-\theta)+\theta)\mathrm{d}x \approx \int_{\mathcal{X}} \alpha(x) \Pr\left[x \in \mathcal{B}\right] \mathrm{d}x \tag{5.17}$$

Hence, the probability $\Pr\left[x \in \mathcal{B}\right]$ is:

$$\Pr\left[x \in \mathcal{B}\right] \approx \pi r_s^2 f_{x_*+\Delta}(x; x_*, \sigma)(\Omega(1-\theta)+\theta) \tag{5.18}$$

The next sections discuss the computation of Ω and θ, respectively.

5.4.1.1 Ω computation

The topology of a WSN cluster can be modeled by means of *random geometric graphs* [PP03] (RGG), since in RGGs vertices (*i.e.* nodes) are connected by an edge depending on the distance between them. In graph theory a set of connected vertices is named *component*. In our cluster deployment scenario it is desirable to obtain the largest component containing the sink. In RGGs (and, generally, in random graphs) there is a limit, called *thermodynamic limit*, related to the node degree which yields to the *giant component* phenomenon. That is, a critical point in the average number of connections per node for which most nodes belong to same component. Reference [PP03] contains analytical descriptions related to the connectivity regime for RGGs with vertices distributed uniformly. Disappointingly, similar results for normally distributed RGGs are still unknown, and the analytical computation of Ω is not possible. To overcome this issue, Ω has been numerically evaluated in this work as a function of N and $\gamma = p_{r_t}\sqrt{N}$. This last parameter is related to the thermodynamic limit in RGGs, and it is commonly used to study connectivity regimes in random graphs. Besides, p_{r_t} is the probability that two nodes are connected, or, in other words, that distance between them is less than r_t. It can be easily calculated (see [VAELMS+07]):

$$p_{r_t} = \Pr\left[d(x_i, x_j)^2 \leq r_t^2\right] = 1 - \exp\left(-\frac{r_t^2}{4\sigma^2}\right) \tag{5.19}$$

Therefore,

$$\gamma = \frac{1 - \exp\left(-\frac{r_t^2}{4\sigma^2}\right)}{\frac{1}{\sqrt{N}}} \tag{5.20}$$

For each configuration of N and γ, 10000 instances of the random network have been selected. For each instance, the order of the component containing the sink has been computed (using Dijkstra's algorithm to determine which nodes have connectivity) obtaining $\frac{\Omega}{N}$, that is, the ratio of connected nodes in the cluster. Finally, the samples from each instance are averaged. We took values ranging from $N = 100$ to $N = 800$, and from $\gamma = 0.1$ to $\gamma = 1/\sqrt{N}$. Results are shown in Figure 5.3. Indeed, since in the optimization model the variable σ (and therefore γ) is continuous, it is desirable to have curves interpolating our numerical evaluations. Then, curves have been fitted using an exponential model:

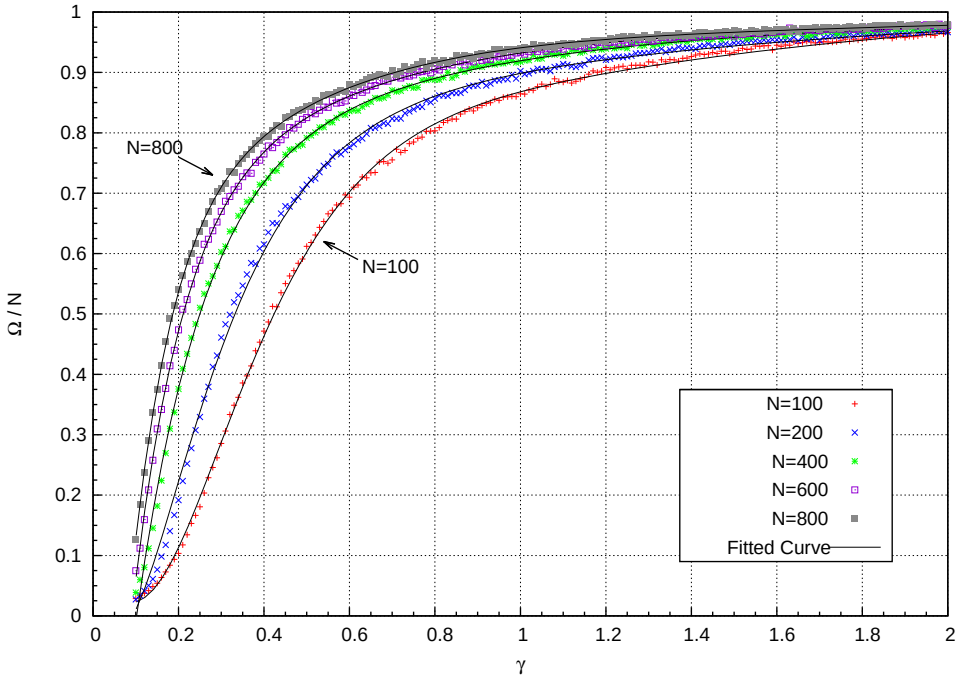


Figure 5.3: $\frac{\Omega}{N}$ curves.

N	a	b	c	d	e	f
100	-1.014	16.54	-14.61	6.367	-1.327	2.517
200	-0.8995	3.189	-7.634	8.334	-0.5282	2.131
400	-1.709	5.891	-0.7896	2.485	-0.02944	1.067
600	-1.846	7.907	-1.455	3.471	-0.09645	1.476
800	-1.898	8.371	-0.1461	1.3	-6.898	5.891

Table 5.1: $\frac{\Omega}{N}$ fitting coefficients of $1 + a\, e^{-b\,\gamma} + c\,\gamma\, e^{-d\,\gamma} + e\,\gamma^2\, e^{-f\,\gamma}$.

$$\frac{\Omega}{N}(\gamma) = 1 + a\, e^{-b\,\gamma} + c\,\gamma\, e^{-d\,\gamma} + e\,\gamma^2\, e^{-f\,\gamma} \tag{5.21}$$

The coefficients which provides the minimal square error fit are shown in Table 5.1.

5.4.1.2 θ computation

Recall that θ represents the ratio of overlapped sensing area in a cluster of N nodes distributed normally around a target point x_*. Clearly, θ does not depend on the central point x_*. Therefore, and without loss of generality, $x_* = (0, 0)$ throughout this section and, thus, $x_i = \Delta$ for all $i = 1, \ldots, N$. Let $\boldsymbol{A}$ denote the area sensed by *all* the nodes in the cluster (which is a random value). Note that the maximum area which can be covered is $N\pi r_s^2$. Therefore,

$$\theta = 1 - \frac{\overline{\boldsymbol{A}}}{N\pi r_s^2} \tag{5.22}$$

In order to compute θ, let $p(x)$ denote the probability that a point $x \in \mathcal{X}$ belongs to the area sensed. Then,

$$\begin{aligned} p(x) &= 1 - \Pr\left[x \notin \cup_{i=1}^{N} B(x_i, r_s)\right] \\ &= 1 - \prod_{i=1}^{N} \left(1 - \Pr\left[x \in B(x_i, r_s)\right]\right) \\ &= 1 - \left(1 - \Pr\left[x \in B(\Delta, r_s)\right]\right)^N \end{aligned} \tag{5.23}$$

Moreover,

$$\Pr\left[x \in B(\Delta, r_s)\right] = \int_{x' \in B(x, r_s)} \frac{1}{2\pi\sigma^2} e^{-\frac{\|x'\|^2}{2\sigma^2}} \mathrm{d}x'$$

For a small $\frac{r_s}{\sigma}$ ratio this equation can be approximated as:

$$\Pr\left[x \in B(\Delta, r_s))\right] \approx \pi r_s^2 f_\Delta(x) \tag{5.24}$$

Where $f_\Delta(x)$ represents the density of probability of the Gaussian random variable Δ in x. Thus, and using the binomial theorem:

$$\begin{aligned} p(x) &\approx 1 - (1 - \pi r_s^2 f_\Delta(x))^N \\ &= 1 - \sum_{i=0}^{N} \binom{N}{i} (-1)^i (\pi r_s^2 f_\Delta(x))^i \\ &= \sum_{i=1}^{N} \binom{N}{i} (-1)^{i+1} (\pi r_s^2 f_\Delta(x))^i \end{aligned} \tag{5.25}$$

Then, the expected covered surface ($\overline{\boldsymbol{A}}$) of the cluster of N nodes is computed by integrating $p(x)$ over $\mathbb{R}^2$:

$$\begin{aligned} \overline{\boldsymbol{A}} &= \int_{x \in \mathbb{R}^2} \sum_{i=1}^{N} \binom{N}{i} (-1)^{i+1} (\pi r_s^2 f_\Delta(x))^i \mathrm{d}x \\ &= \sum_{i=1}^{N} \binom{N}{i} (-1)^{i+1} \int_{x \in \mathbb{R}^2} \frac{(\pi r_s^2)^i}{(2\pi\sigma^2)^i} e^{-\frac{i(\|x\|^2)}{2\sigma^2}} \mathrm{d}x \\ &= \sum_{i=1}^{N} \binom{N}{i} (-1)^{i+1} \frac{(r_s^2)^i}{(2\sigma^2)^{i-1}} \frac{\pi}{i} \end{aligned} \tag{5.26}$$

Hence, θ is given by:

$$\theta = 1 - \frac{\overline{\boldsymbol{A}}}{N\pi r_s^2} = 1 - \sum_{i=1}^{N} \binom{N}{i} (-1)^{i+1} \left(\frac{r_s^2}{2\sigma^2}\right)^{i-1} \frac{1}{i}\frac{1}{N} \tag{5.27}$$

5.4.2 Statement of the global optimization

Therefore, from (5.11), (5.18), and (5.4) the global problem can be stated as solving:

$$\max_{\substack{x_*^j,\sigma_j \\ j=1,\ldots,M}} \sum_{j=1}^{M} \int_{\mathcal{X}} \mathbb{E}\left[\alpha^j(x)\right] \pi r_s^2 f_{x_*^j+\Delta_j}(x; x_*^j, \sigma_j)(\Omega_j(1-\theta_j)+\theta_j)\mathrm{d}x \tag{5.28}$$

where

$$\mathbb{E}\left[\alpha^j(x)\right] = \alpha(x) \prod_{k=1}^{j-1} (1 - \pi r_s^2 f_{x_*^k+\Delta_k}(x; x_*^k, \sigma_k)(\Omega_k(1-\theta_k)+\theta_k))$$

where Ω and θ are given respectively by (5.21) and (5.27).
And subject to

$$x_*^j \in \mathcal{X}$$
$$\sigma_j > 0$$

for $j = 1, \ldots, M$.

This global version of the problem involves the *simultaneous* computation of all the variables. As will be shown in Section 5.5 this approach does not scale up well in memory. An alternative approach is to solve each cluster deployment separately. That is, after selecting a solution for a cluster this solution does not change. This leads to the iterative optimization version, described in the following section.

5.4.3 Statement of the iterative optimization

In order to avoid scalability issues the problem can be divided into a sequence of steps; at each step, the optimization of just a single cluster if considered. That is, at step j only the cluster j configuration x_*^j, σ_j is selected. Once fixed, the configuration will not vary for the next steps. At each step the map needs to be corrected to discount the effect of previous clusters. This can be done easily, since $\mathbb{E}\left[\alpha^j(x)\right]$ is the expected map with the remaining importance.

Thus, the following iterative optimization procedure is used:

1. Set $\mathbb{E}\left[\alpha^1(x)\right] = \alpha(x)$
2. For $j = 1, \ldots, M$

(a) Given the importance map $\mathbb{E}\left[\alpha^1(x)\right]$, solve the optimization problem:

$$\max_{x_*^j,\sigma_j} \int_{\mathcal{X}} \mathbb{E}\left[\alpha^j(x)\right] \pi r_s^2 f_{x_*^j+\Delta_j}(x; x_*^j, \sigma_j)(\Omega_j(1-\theta_j)+\theta_j)\mathrm{d}x \tag{5.29}$$

subject to

$$x_*^j \in \mathcal{X}$$
$$\sigma_j > 0$$

(b) Compute

$$\mathbb{E}\left[\alpha^{j+1}(x)\right] = \mathbb{E}\left[\alpha^j(x)\right]\left(1 - \pi r_s^2 f_{x_*^j+\Delta_j}(x; x_*^j, \sigma_j)(\Omega_j(1-\theta_j)+\theta_j)\right)$$

as the updated importance map for step $j+1$.

This leads to:

$$\sum_{j=1}^{M} \max_{x_*^j,\sigma_j} \int_{\mathcal{X}} \alpha^j(x) \pi r_s^2 f_{x_*^j+\Delta_j}(x; x_*^j, \sigma_j)(\Omega_j(1-\theta_j)+\theta_j)\mathrm{d}x \tag{5.30}$$

Note that here the sum sign is outside the optimization operator, unlike the situation in formula (5.28). Therefore this approach may not achieve globally optimal solutions. Instead, iterative optimization only finds optimal cluster configurations for a given step j. The rationale is that having a cluster capture high importance at a given step may lead to "bad" clusters capturing low importance in subsequent steps. This is particularly true if the area to cover is small compared to the size of the cluster. Section 5.5 provides examples of this kind.

However, in scenarios in which the area is much larger than the size of the cluster, this effect is less noticeable since the best solution tends to separate clusters because areas already covered possess less importance. This effect is also shown in the examples provided in Section 5.5. Besides, the iterative optimization version has the advantage of being scalable to any number of clusters.

5.5 Deployment examples and results

The main scenario selected to test the performance of our optimization methodology is the lunar crater Tycho (Figure 5.4). Tycho is a prominent lunar impact crater located in the southern lunar highlands. Figure 5.4(b) illustrates an image of Tycho (35×100 Km^2) that the Clementine mission [CLE11] mapped onto the Moon's surface. The resulting size of the image was 232×1100 pixels. This color scale mosaic allows different rock units to be distinguished and shows the location of fresh, "mafic" material (*i.e.*, materials relatively rich in iron and magnesium). The importance selected ranges from 0 (minimum) to 255 (maximum), and is related to the presence of these materials.

		Γ(%)	
		Analytical	Experimental
N=800	$r_s = 4,\ r_t = 12$	53.2%	49.1%
N=600	$r_s = 4,\ r_t = 12$	44.4%	41.4%
N=400	$r_s = 4,\ r_t = 12$	33.8%	31.5%
N=200	$r_s = 4,\ r_t = 12$	20.5%	19.4%
N=100	$r_s = 4,\ r_t = 12$	12.3%	11.8%

Table 5.2: Analytical vs experimental values computed averaging 5000 experiments, Tycho map for $M = 10$.

Let us remark that the optimization procedure is *independent* of the importance map selected and of the network parameters N, r_t and r_s. Therefore, no changes are required to compute the best placement for a different configuration.

Our optimization algorithm has been implemented using the *general algebraic modeling system* (GAMS) [BKM+98], a high-level modeling system for mathematical programming and optimization, and CONOPT [Dru85], a solver for large-scale nonlinear optimization. GAMS allowed us to define our optimization problem directly from the mathematical descriptions provided in Sections 5.4.2 and 5.4.3.

Let us recall that in the global optimization case the GAMS algorithm is executed to simultaneously find the optimal target point (x_*) and dispersion (σ) for the M clusters (see Section 5.4.2), whereas for the iterative case a GAMS instance is consecutively run M times using modified importance maps (see Section 5.4.3). In both cases, the CONOPT solver proceeds iteratively until the algorithm eventually converges to an optimum. Besides, the solver requires initial values for the variables x_* and σ (M initial pairs in the global case) to start computations, and the final solution is not guaranteed to be a global optimum solution, but rather a local one. We observed large variations in the solution depending on the initial positions x_*. To mitigate this we ran each optimization problem with several initial values and selected the best one. In all the examples in this section the global method considers 75 randomly chosen initial cluster placements. In the iterative method we fixed 36 different initial x_* per step, as shown in Figure 5.4(c). Let us remark that the overall number of runs in the iterative case is then $M \times 36$, and therefore for $M > 2$ the iterative method requires more runs. Despite this, it is scalable and requires much less time than the global method, as will be shown later in Section 5.5.2.

5.5.1 Validation tests

Our first aim was to validate the approaches taken in Section 5.4.1 related to the single cluster coverage computation. Several tests were performed comparing the importance expected by the analysis with the experimental value obtained by averaging random instances of the network using the optimal parameters selected by the solver (*e.g.* see Figure 5.5). Table 5.2 summarizes the comparison between the analysis and the experimental procedure (averaging the result of 5000 experiments). The ratio of importance $\Gamma_{\%}$ is rela-

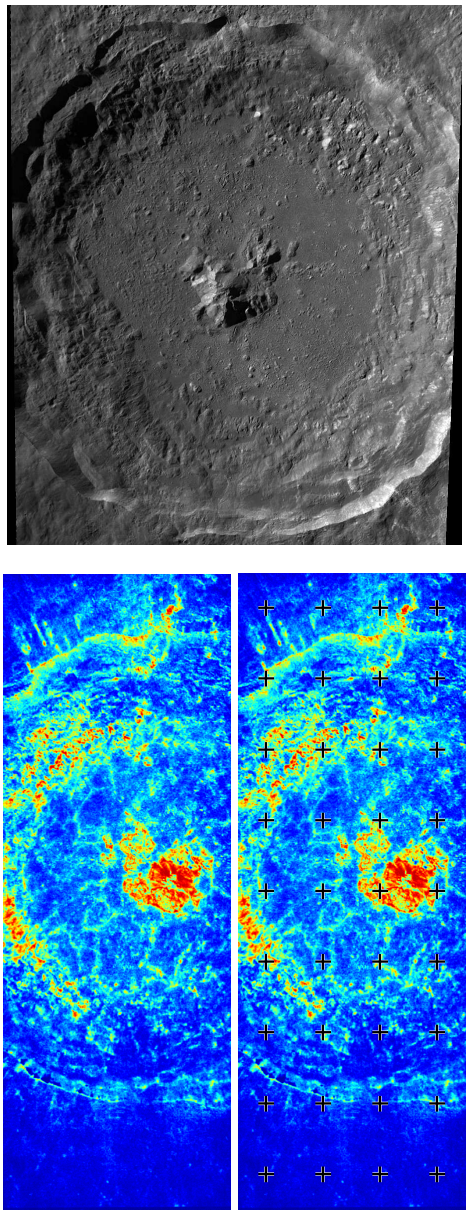
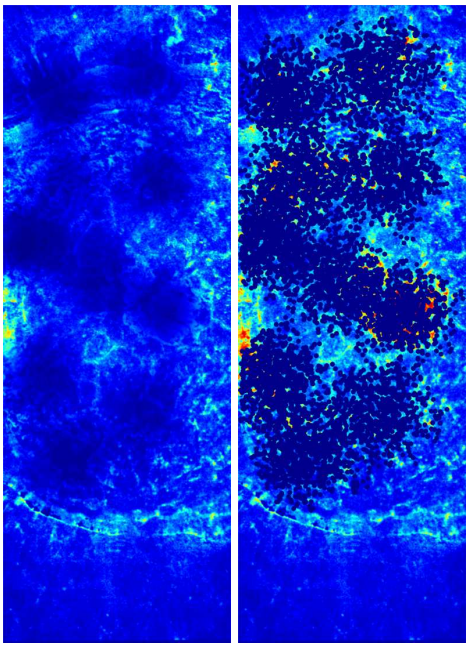


Figure 5.4: (a) LROC WAC image of Tycho crater [NASA/GSFC/Arizona State University], (b) Tycho image from UV/visible camera on the Clementine spacecraft showing the location of fresh, "mafic" material, (c) Initial levels for x_* variable marked with '+', iterative optimization.



(a) Analytic (b) Experimental

Figure 5.5: Analytic vs experimental multi-deployment Tycho map for $M = 10$, $N = 800$, $r_s = 4$, $r_t = 12$. Dark blue areas indicate the locations covered.

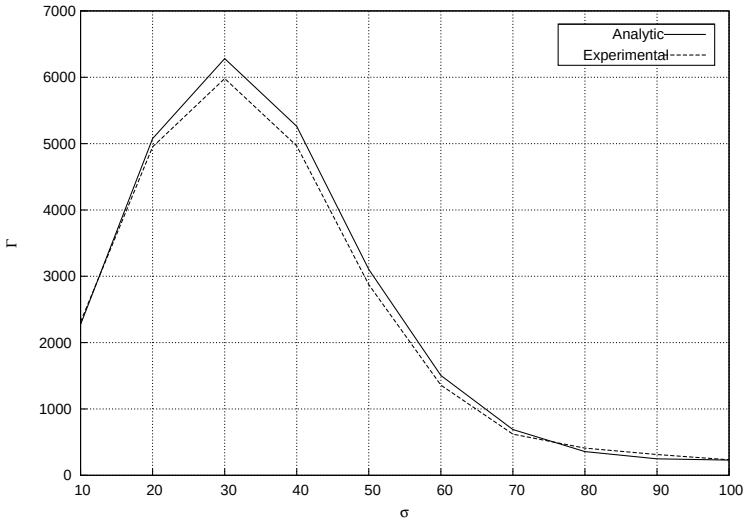
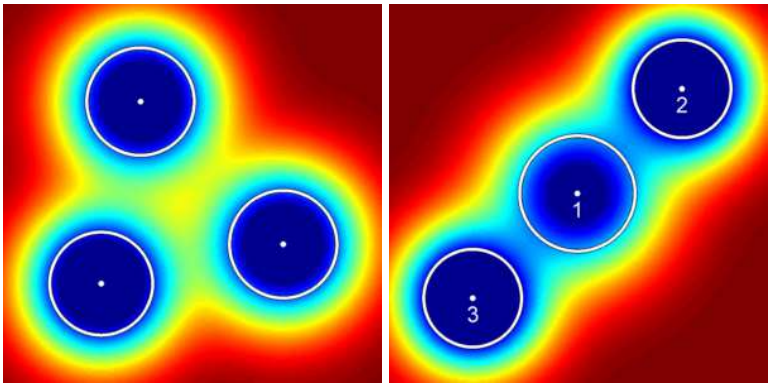


Figure 5.6: Analytic versus experimental importance, homogeneous map, $N = 100$ nodes.



(a) Global (b) Iterative

Figure 5.7: Multi-deployment solution, global method, homogeneous map, for $M = 3$, $N = 600$, $r_s = 4$, $r_t = 12$. Red intensities indicate areas not covered, blue ones indicate areas covered.

tive to the total importance of the map. These results indicate that the importance captured is derived with less than 4.1% deviation in the worst case, despite the approximations taken in the analysis.

An additional test was performed using the constant importance map $\alpha(x) = 1$ for all $x \in \mathcal{X}$. In this experiment a cluster of $N = 100$ nodes was aimed at the center of coordinates, with a variable dispersion σ ranging from 10 to 100. The importance covered was computed analytically using (5.14) and experimentally averaging the results of 5000 random deployments. The results are shown in Figure 5.6. The divergence is below 3% in the worst case, and more remarkable is the fact that the optimum was reached for the same dispersion. This shows that, even though there exists a small divergence in the predicted importance, the parameters computed for the cluster are optimal.

In the next sections all the importances shown were computed experimentally by averaging 5000 samples using the cluster parameters obtained using the optimization procedure; therefore, these values represent the real importances.

5.5.2 Global versus Iterative solution

These tests aimed at comparing the performance of the global and iterative algorithms. As stated in Sections 5.4.2 and 5.4.3, if the size of the map is comparable to the size of the cluster the global optimization algorithm should perform better. This can be checked in the example shown in Figure 5.7. The importance map is a 300×300 area with $\alpha(x) = 1$ for all $x \in \mathcal{X}$. The global solution covers 42.7% of the whole importance, and the iterative one 38.8%. Besides, the first cluster in the iterative case covered 15.3% of the importance, whereas the best cluster in the global case covered only 14.7%. This example shows that it is better to have equilibrated clusters (global case), rather than a good cluster that prevents good placement of the subsequent clusters (iterative case).

However, if the map is larger, this effect is less noticeable. Both optimization strategies were applied to the Tycho map (see Table 5.3 for a summary of the results). Both opti-

	Iterative			Global		
M	$\Gamma(\%)$	Time (m:s)	Memory (Mb)	$\Gamma(\%)$	Time (m:s)	Memory (Mb)
1	0.07	08:22	577	0.07	17:02	577
2	0.13	16:44	577	0.13	77:28	1736
3	0.18	25:06	577	0.18	193:18	3526
4	0.22	33:28	577	0.22	369:36	5946
5	0.27	41:50	577	0.27	594:00	8995

Table 5.3: Performance of iterative vs global methods, Tycho map, $N = 600$, $r_s = 4$, $r_t = 12$ (CPU Intel Core i7 2600K).

mization methods were able to cover the same importance. However, the global method required more computing resources (both CPU time and memory). In fact, for more than 5 clusters the GAMS/CONOPT global implementation was unable to reach a solution due to the lack of computational resources.

Summarizing, the iterative method is scalable and achieves similar results to the global method in scenarios in which the target area is much larger than the size of the cluster. Since most real scenarios will be large and require a large number of clusters, the iterative method must be selected. The next sections analyze the performance of this method in depth.

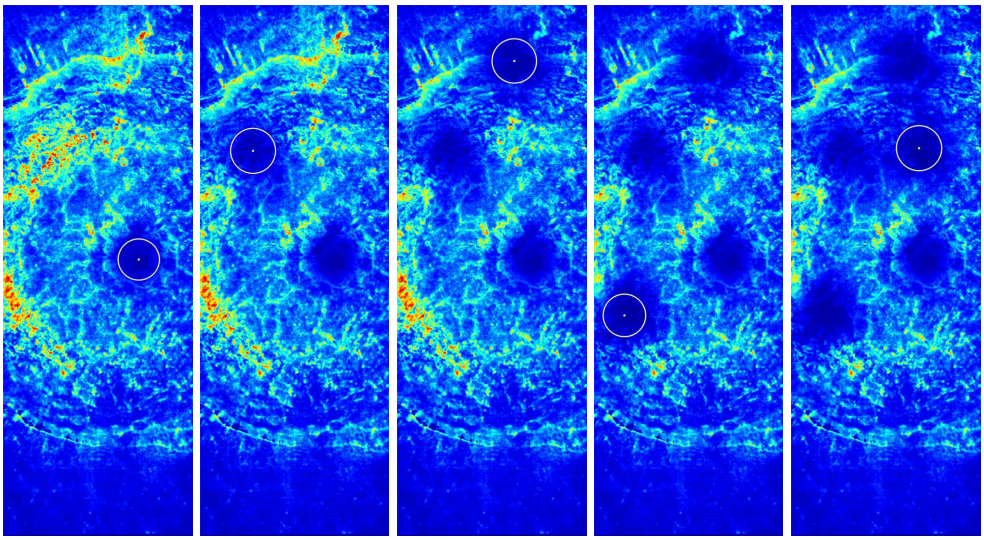
5.5.3 An iterative optimization example: The Tycho crater

In these tests we selected $M = 10$, $N = 800$, and three combinations of sensing/ communication ranges: ($r_s = 2$, $r_t = 6$), ($r_s = 4$, $r_t = 12$), and ($r_s = 6$, $r_t = 18$). As an example of the optimization process for the configuration ($r_s = 4$, $r_t = 12$) Figure 5.8 shows one frame per cluster. Frame j represents the importance that remains after the step, *i.e.* $\mathbb{E}\left[\alpha^{j+1}(x)\right]$. In each frame the point depicts x_* and the dispersion σ is marked with a circle. The solution found with our algorithm covers almost all of the important zones of the map.

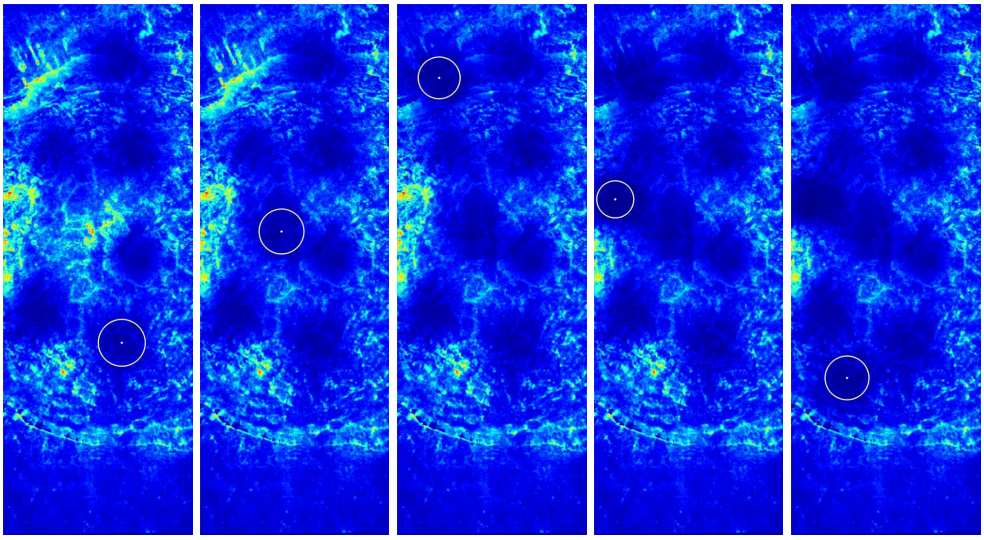
Figure 5.9 shows frames summarizing the optimal solution for each configuration of (r_s, r_t). Besides, the cumulative ratio of importance $\Gamma_{\%}$ per cluster relative to the total importance is shown in Figure 5.10. Note how the configuration of the ranges drastically affects the solution. There is almost no variation in σ for the low ranges whereas there is noticeable variation for mid and high ranges. As can be seen in the figures, the importance captured per cluster decreases as the steps increase. This is also an expected effect since the first clusters cover the best sites.

In addition, elapsed computation times on an Intel Core i7 2600K were 30 minutes per cluster placement, hence it took only 5 hours for the optimization of a multi-deployment of M=10 clusters. Therefore, the speed of the optimization algorithm is notably fast.

Note that a single cluster captures only a small ratio of the whole region importance (3%, 9%, and 16% depending on the sensing range) for $N = 800$. If the number of nodes

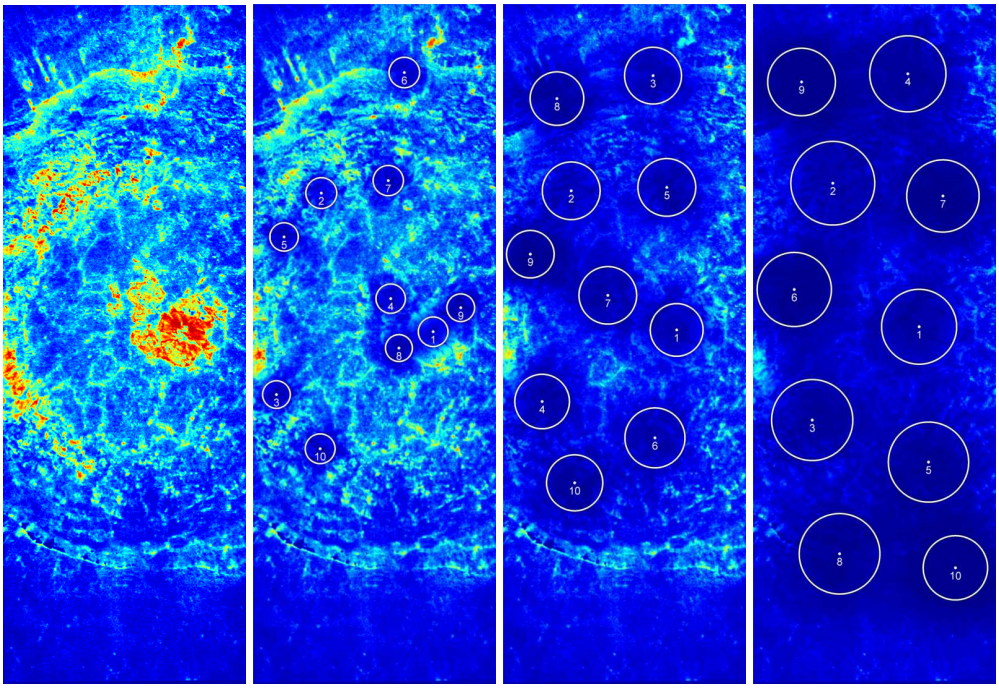


(a) $j = 1$ (b) $j = 2$ (c) $j = 3$ (d) $j = 4$ (e) $j = 5$



(f) $j = 6$ (g) $j = 7$ (h) $j = 8$ (i) $j = 9$ (j) $j = 10$

Figure 5.8: Multi-deployment solution for Tycho map, $M = 10$, $N = 800$, $r_s = 4$, $r_t = 12$.



(a) Original (b) $r_s = 2, r_t = 6$ (c) $r_s = 4, r_t = 12$ (d) $r_s = 6, r_t = 18$

Figure 5.9: Final multi-deployment solution for Tycho map, $M = 10$, $N = 800$.

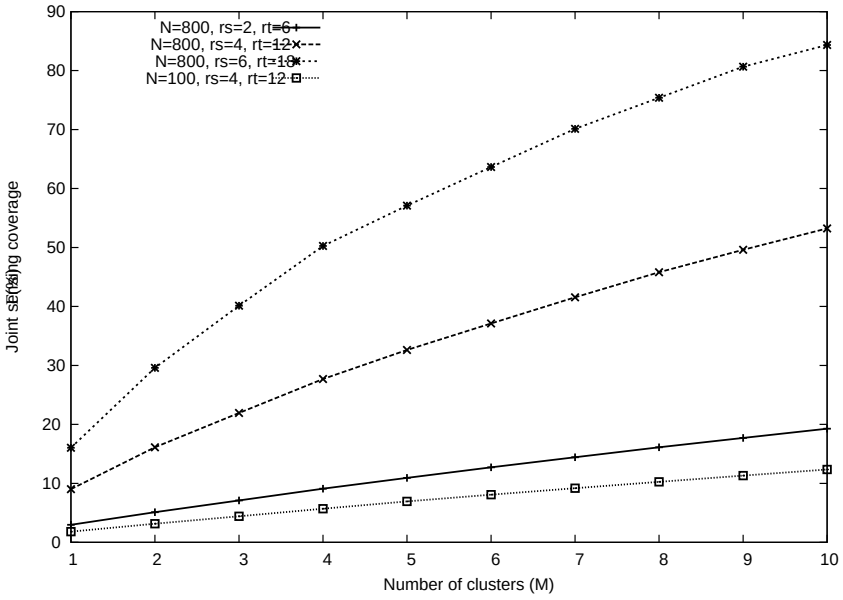


Figure 5.10: Joint sensing coverage $\Gamma(\%)$ versus M, Tycho map.

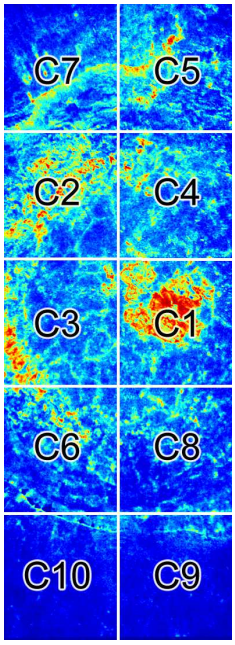


Figure 5.11: Cell division for the heuristic placement, Tycho map.

is smaller, this effect is more noticeable, *e.g.* for $N = 100$ with $r_s = 4$ (see Figure 5.10) a single cluster collects only 1.82% of the whole importance. Clearly, to cover large areas either a cluster needs to be composed of a very large number of nodes or several clusters are required. The former approach may be unfeasible if the number of nodes is too high, and, more important, our results indicate that several clusters achieve better sensing coverage than a single cluster with the same number of nodes. For example, with $N = 800$, a single cluster (let us assume $r_s = 4$) can collect 9% of the whole importance, but 8 clusters of $N = 100$ are able to collect 10.78%.

5.5.4 Iterative optimization versus heuristic placement

In the final tests, the optimal strategy computed with our algorithm is compared against a general heuristic based on related works [LRS09, ZC03] which divide the target area into a grid and use one cluster per cell. Following the results in [LRS09], the optimal target point for the cluster must be the center of the cell. In the implemented heuristic we follow this approach by: (i) dividing the target area into $M = 10$ equally spaced cells (Figure 5.11), (ii) aiming the cluster at the center of the cell and selecting σ values 25, 50, 75, and 100. The solution computed with our optimal approach outperforms the heuristic in all cases. The improvement percentages are reported in Table 5.4.

Even selecting the best σ parameter in the heuristic approach (we must remark that previous works do not provide methods to compute this) our approach improves the result by more than 14% in the worst case ($N = 800$). In a scenario with fewer nodes, where the optimization problem is more challenging, the optimal approach developed clearly outperforms the heuristic (*e.g.* by more than 82% for $N = 100$). The difference may even greater if σ is poorly selected (*e.g.* more than 400% in all cases if $\sigma = 100$).

	$\sigma = 25$	$\sigma = 50$	$\sigma = 75$	$\sigma = 100$
$N = 100$	$\mathbf{82,59}\%$	$1989,28\%$	$5624,79\%$	$8554,04\%$
$N = 200$	$\mathbf{47,71}\%$	$369,96\%$	$4116,63\%$	$9345,81\%$
$N = 400$	$\mathbf{53,76}\%$	$54,42\%$	$643,22\%$	$5626,80\%$
$N = 600$	$67,35\%$	$\mathbf{24,41}\%$	$132,55\%$	$1680,51\%$
$N = 800$	$77,07\%$	$\mathbf{14,24}\%$	$56,48\%$	$401,79\%$

Table 5.4: Improvement in terms of covered importance of the optimal solution compared with the heuristic. Tycho map, $M = 10$, $N = 800$, $r_s = 4$, $r_t = 12$.

5.6 Summary

In this chapter we have developed mathematical and algorithmic procedures to calculate the best placement options for WSN Gaussian clusters in an area of non-homogeneous importance. In the formulation of the optimization problem we have captured the main characteristics of the WSN model and made realistic assumptions: the clusters must follow a normal distribution over the monitored area and must also cover the best areas. At the same time, they must be connected, *i.e.* nodes cannot be isolated.

We have shown that the approximations selected in our analytical procedure achieve notably good results, closely resembling those obtained in experimental evaluations. Besides, as an important conclusion, in a single cluster optimal deployment, hundreds (if not thousands) of sensors are required to completely sense some relative "small" regions. Iterative multi-deployment proves to be an effective strategy to overcome this problem. Results demonstrate how large non-homogeneous regions may be covered with several clusters using suitable configurations.

Chapter 6

Conclusions and future works

In this chapter, we summarize the main results of the thesis. We also propose possible future lines of research in this area of our work.

6.1 Conclusions

The first important conclusion is drawn from the research challenges presented by AmI environments (Chapter 1). As we have shown, AmI environments are extremely beneficial in numerous everyday human activities (traveling, working, practicing sports, etc.). Yet these are complex systems whose development is influenced by a considerable number of factors, including learning and decision-making processes, which are vital. This thesis demonstrates how to use three classification and inference methods in decision making: k-NN, (m, s)-splines and Markov decision processes (MDP). These techniques provide users with rapid and useful responses, by adapting to changing user needs.

Moreover, the quantity and quality of the information supplied directly influence decision making. Sensors must be carefully placed in areas to ensure that only the most relevant data are captured. Optimal planning for network deployment will help us determine the position of each node in order to maximize the importance of the data captured. In this context, this thesis outlines how to undertake two different types of deployment (deterministic and random). For this kind of problem, we use three novel methodologies to plan node position: $\text{ACO}_{\mathbb{R}}$, gradient descent method and a partially-controlled-clustered methodology. In the third case, the technique is highly original and was especially developed as part of this thesis to perform random deployment of sensors over large surfaces.

Finally, this thesis presents two innovative ambient intelligence applications: a scenario to aid athletes in their training, and another for detecting poaching activities.

6.1.1 Decision making

In this thesis we show that AmI environments help athletes in the pursuit of their sport. Specifically, in Chapter 2, we have developed a system capable of advising runners on their training. For that purpose, we deployed a system over a cross-country running circuit. A network of sensors constantly monitors runners' location, biometric parameters

(heart rate) and environment (temperature).

A control node (CN), where we have located the *intelligence* of the system, gathers data monitored by the network. This node computes the best training option (*i.e.* the path to follow at a track fork). For analysis, it uses the current monitored data and the record of decisions taken by the runner. In the same conditions (Chapter 3), we tested decision methods for two different training objectives.

The techniques of k-NN and (m, s)-splines compute the best option to successfully complete only the following training period (single-step decision making). Although not perfect, these classification methods achieve a ratio success in the range of 70% (k-NN) to 85% (splines interpolation).

With MDP the system maximizes the performance for all future training steps, rather than considering only a horizon with a single step of decision (multi-step decision-making). In this case, MDP achieves a 70% match in an homogeneous training program. Note that this is similar to the k-NN classification ratio. However both results are not directly comparable since k-NN success ratio includes the out-of-range events, whereas MDP do not. Therefore MDP results are more impressive.

Furthermore, a fundamental outcome of the analyses undertaken in this thesis is that environmental data improve the success ratio of all these methodologies.

6.1.2 Node placement

With the aim of maximizing the quality and quantity of information captured by sensor nodes, this thesis proposes several innovative sensor positioning techniques.

In deterministic deployment conditions (see Chapter 4), we used the $\text{ACO}_{\mathbb{R}}$ metaheuristic for this type of problem for the first time. We carried out tests in two different deployment scenarios (convex and non-convex) in order to study performance. Both scenarios pose an important challenge for the positioning methodology since it requires nodes to be placed in regions where the importance of data capture is relatively low in order to achieve regions of high importance. Captured importance with $\text{ACO}_{\mathbb{R}}$ is greater than that obtained with previous heuristics. Moreover, we confirmed that $\text{ACO}_{\mathbb{R}}$ scales well with the size of the problem (number of nodes and size of deployment scenario).

In Chapter 4 , we develop an AmI system that detects gunshot location based on information supplied by a network of sensors. In this case network deployment is multi-objective. On the one hand, the nodes must be positioned to maximize accurate detection. On the other, we must place the nodes near power lines to reduce cost. That is, the objective is two-fold: to achieve the highest quantity and quality of information, while minimizing deployment costs. In this case a Pareto frontier approach is selected to characterize solutions of large-scale deployments. We compute the position of the nodes by using a monotone descent method without derivatives that is compatible with realistic optimization functions. The proposed model achieves 85% coverage with 50% reduction in deployment costs. These results are clearly superior to unguided placement.

Furthermore, the AmI system developed for gunshot detection is entirely new. The location method used is hyperbolic multilateration, which is based on clock information from each node. Real-time detection means that the clocks must always be synchronized. To that end, in this thesis, we develop a new time synchronization algorithm. The advan-

tage of this algorithm is that it minimizes the number of messages exchanged between nodes. This enhances speed but also reduces power consumption during the synchronization process. Finally, tests performed in two-dimensional scenarios conclude that at least four nodes are required for satisfactory gunshot location accuracy.

In the area of random deployment, this thesis develops a novel methodology that enables the monitoring of large surfaces by using a number of WSN clusters (Chapter 5). The optimization model assumes the realistic procedure of network deployment by scattering several clusters of nodes from an aircraft. The optimization problem considers Gaussian distribution for each cluster whose dispersion is determined by the release altitude. Although the exact position of the nodes is unknown, the method successfully ensures that most of the nodes in each cluster are connected. Connectivity is a requirement since areas monitored by isolated nodes do not contribute to the information captured by the network. Experimental evaluations demonstrate that the approximations selected in our analytical procedure achieve notably good results, closely resembling those obtained in experimental evaluations. Besides, as an important conclusion, in a single cluster optimal deployment hundreds (if not thousands) of sensors are required to completely sense some relatively 'small' regions. Iterative multi-deployment proves to be an effective strategy to overcome this problem.

6.1.3 Future works

Throughout this thesis we have developed and evaluated innovative AmI systems, while assessing several decision-making techniques, with promising results. We have also completed several rigorous studies on sensor positioning methods.

The creation of new AmI systems in environments where their use would be advantageous to users is a natural continuation of this work. Decision-making systems should be developed within these new environments, using techniques similar to those explained here or by exploring other mathematical methods, for example, the branch of decision or game theory.

A more realistic modeling of the deployment area would also be useful as part of the study of optimal positioning techniques, since it would lead to results that better reflected reality. For instance, by considering the placement of sensors in three dimensions and their importance as a time evolving magnitude, namely, a stochastic process.

Finally, we would like to reflect on the tendency that AmI systems may follow. In recent years, construction techniques for MEMS devices have continually evolved. This will lead to cheaper devices, with greater sensing capabilities, whose smaller size will allow for seamless integration into everyday objects. Improved manufacturing techniques will therefore facilitate the construction of less intrusive ambient intelligence systems. Furthermore, more efficient communication systems (faster, with lower power consumption) are also being developed. In all likelihood, these advances will lead to the development of more user-friendly user interfaces. In the light of the studies undertaken in this thesis, we shall then observe how technology evolves, so that we may adapt these future improvements to the construction of AmI systems.

Bibliography

[AAA09] M. Arif, M.U. Akram, and F.A. Afsar. Arrhythmia beat classification using pruned fuzzy k-nearest neighbor classifier. In *Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of*, pages 37–42. IEEE, 2009.

[AABP10] N. Aitsaadi, N. Achir, K. Boussetta, and G. Pujolle. Multi-objective WSN deployment: Quality of monitoring, connectivity and lifetime. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6. IEEE, 2010.

[AABP11] N. Aitsaadi, N. Achir, K. Boussetta, and G. Pujolle. Artificial potential field approach in WSN deployment: Cost, QoM, connectivity, and lifetime constraints. *Computer Networks*, 55(1):84–105, 2011.

[AB08] D.K. Arvind and A. Bates. The speckled golfer. In *Proceedings of the ICST 3rd international conference on Body area networks*, page 28. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[AD08] H.M. Ammari and S.K. Das. Integrated coverage and connectivity in wireless sensor networks: A two-dimensional percolation problem. *IEEE Transactions on Computers*, 57(10):1423–1434, 2008.

[AJK07] S. Armstrong, E. Jovanov, and D.G. Kerwin. Wireless connectivity for health and sports monitoring: a review. *British Medical Journal*, 41(5):285, 2007.

[AM94] H.R. Anderson and J.P. McGeehan. Optimizing microcell base station locations using simulated annealing techniques. In *Vehicular Technology Conference, 1994 IEEE 44th*, pages 858–862. IEEE, 1994.

[ASBG99] P.A. Absil, R. Sepulchre, A. Bilge, and P. Gérard. Nonlinear analysis of cardiac rhythm fluctuations using DFA method. *Physica-Section A*, 272(1-2):235–244, 1999.

[AT04] R. Almgren and A. Tourin. Optimal soaring with Hamilton-Jacobi-Bellman equations, 2004. http://www.cims.nyu.edu/~almgren/optsoar/optsoar.pdf.

[AY08] K. Akkaya and M. Younis. Coverage and latency aware actor placement mechanisms in WSANs. *International Journal of Sensor Networks*, 3(3):152–164, 2008.

[AYY07] K. Akkaya, M. Younis, and W. Youssef. Positioning of base stations in wireless sensor networks. *IEEE Communications Magazine*, 45(4):96–102, 2007.

[BAS97] M.S. Brandstein, J.E. Adcock, and H.F. Silverman. A closed-form location estimator for use with room environment microphone arrays. *Speech and Audio Processing, IEEE Transactions on*, 5(1):45–50, 1997.

[BCR+11] A. Bari, Y. Chen, D. Roy, A. Jaekel, and S. Bandyopadhyay. Designing hierarchical sensor networks with mobile data collectors. *Pervasive and Mobile Computing*, 7(1):128–139, 2011.

[BD62] R.E. Bellman and S.E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, 1962.

[Bis06] C.M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[BJB08] A. Bari, A. Jaekel, and S. Bandyopadhyay. Clustering strategies for improving the lifetime of two-tiered sensor networks. *Computer Communications*, 31(14):3451–3459, 2008.

[BJM04] P. Brockhausen, T. Joachims, and K. Morik. Combining statistical learning with a knowledge-based approach. *HT014602036*, 2004.

[BK06] A. Baca and P. Kornfeind. Rapid feedback systems for elite sports training. *Pervasive Computing, IEEE*, 5(4):70–76, Oct 2006.

[BKM+98] A. Brooke, D. Kendrick, A. Meeraus, R. Raman, and US America. The general algebraic modeling system. *GAMS Development Corporation*, 1998.

[BKP+10] A. Baca, P. Kornfeind, E. Preuschl, S. Bichler, M. Tampier, and H. Novatchkov. A server-based mobile coaching system. *Sensors*, 10(12):10640–10662, 2010.

[Buc06] V. Bucur. *Acoustics of wood*, volume 1431. Springer Verlag, 2006.

[BvSNP03] K. Backx, K.A. van Someren, A.M. Nevill, and G.S. Palmer. Mathematical prediction of one hour cycle time trial performance under different ambient temperatures. *Medicine & Science in Sports & Exercise*, 35(5):S30, 2003.

[CCZ05] Y. Chen, C.N. Chuah, and Q. Zhao. Sensor placement for maximizing lifetime per unit cost in wireless sensor networks. In *Military Communications Conference, 2005. MILCOM 2005. IEEE*, pages 1097 –1102 Vol. 2, oct. 2005.

[CD07] P. Cunningham and S.J. Delany. k-Nearest Neighbour Classifiers. *Multiple Classifier Systems*, pages 1–17, 2007.

[CDWX08] X. Cheng, D.Z. Du, L. Wang, and B. Xu. Relay sensor placement in wireless sensor networks. *Wireless Networks*, 14(3):347–355, 2008.

[CER05] C. Clavel, T. Ehrette, and G. Richard. Events detection for an audio-based surveillance system. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1306–1309. IEEE, 2005.

[Chi05] E.H. Chi. Introducing wearable force sensors in martial arts. *IEEE Pervasive Computing*, 4(3):47–53, 2005.

[Cla88] S.R. Clarke. Dynamic programming in one-day cricket-optimal scoring rates. *Journal of the Operational Research Society*, 39(4):331–337, 1988.

[CLE11] Clementine project information, October 2011. http://nssdc.gsfc.nasa.gov/planetary/clementine.html.

[Croa] Crossbow Inc. Crossbow sensors technology. http://www.xbow.com/.

[Crob] Crossbow Inc. IMB400 -Imote2 Multimedia Board. http://www.openautomation.net/page/productos/id/29/title/IMB-Multimedia-Board.

[Croc] Crossbow Inc. Imote2 -High-performance Wireless Sensor Network Node. Hardware Reference Manual. http://www.xbow.com.cn/LinkClick.aspx?fileticket=i7FK6%2B1oodU%3D&tabid=121.

[Crod] Crossbow Inc. MICAz -Wireless measurement system. http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf.

[Croe] Crossbow Inc. MTS400 -Mote Processor Radio boards. http://www.iwi.uni-hannover.de/lv/seminar_ss04/www/Jan_Gacnik/bibliography/XBOW-MTS400.pdf.

[CYE+03] J.C. Chen, L. Yip, J. Elson, H. Wang, D. Maniezzo, R.E. Hudson, K. Yao, and D. Estrin. Coherent acoustic array processing and localization on wireless sensor networks. *Proceedings of the IEEE*, 91(8):1154–1162, 2003.

[D. 01] D. Estrin et al. Embedded everywhere: A research agenda for networked systems of embedded computers. *Computer Science and Telecommunications Board (CSTB) Report*, 2001.

[DBS+01] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J-C. Burgelman. Scenarios for ambient intelligence in 2010. Technical report, Information Society Technologies Advisory Group, 2001.

[DC03] S.S. Dhillon and K. Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *Proc. of IEEE Wireless Communications and Networking Conference*, 2003.

[DCI02] S.S. Dhillon, K. Chakrabarty, and SS Iyengar. Sensor placement for grid coverage under imprecise detections. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 2, pages 1581–1587. IEEE, 2002.

[Def06] Defense Review. Anti-sniper/ sniper detection/ gunfire detection systems at a glance, July 2006. http://www.defensereview.com/anti-snipersniper-detectiongunfire-detection-systems-at-a-glance/.

[DIL+05] S. Dubowsky, K. Iagnemma, S. Liberatore, DM Lambeth, JS Plante, and PJ Boston. A concept mission: Microbots for large-scale planetary surface and subsurface exploration. In *AIP Conference Proceedings*, volume 746, page 1449, 2005.

[DKK03] K. Dasgupta, M. Kukreja, and K. Kalpakis. Topology-aware placement and role assignment for energy-efficient information gathering in sensor networks. In *Computers and Communication, 2003.(ISCC 2003). Proceedings. 8-th IEEE International Symposium on*, pages 341–348. IEEE, 2003.

[Dor92] M. Dorigo. Optimization, learning and natural algorithms. *Ph.D. Thesis, Politecnico di Milano, Italy*, 1992.

[Dru85] A. Drud. CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31(2):153–191, 1985.

[DSO08] E. Doukhnitch, M. Salamah, and E. Ozen. An efficient approach for trilateration in 3D positioning. *Computer Communications*, 31(17):4124–4129, 2008.

[EGE02] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36(SI):147–163, 2002.

[EHPM05] A. Efrat, S. Har-Peled, and J.S.B. Mitchell. Approximation algorithms for two optimal location problems in sensor networks. In *Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on*, pages 714–723. IEEE, 2005.

[ELP] ELPAÍS.com. Detenidos 12 cazadores furtivos en la Operación Bambi. http://elpais.com/elpais/2008/03/12/actualidad/1205313430_850215.html.

[FDN05] L. Fang, W. Du, and P. Ning. A beacon-less location discovery scheme for wireless sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 161–171. IEEE, 2005.

[FH89] E. Fix and J.L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 57(3):238–247, 1989.

[FHT01] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001.

[For04] L.J. Fortier. *An Application of a Proposed Airdrop Planning System*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 2004.

[Gag92] D.W. Gage. Command control for many-robot systems. Technical report, DTIC Document, 1992.

[GBGL08] T. Gowers, J. Barrow-Green, and I. Leader. *The Princeton companion to mathematics*. Princeton University Press, 2008.

[GCACM+09] F.J. González-Castaño, J. Vales Alonso, E. Costa-Montenegro, P. López-Matencio, F. Vicente-Carrasco, F.J. Parrado-García, F. Gil-Castiñeira, and S. Costas-Rodríguez. Acoustic sensor planning for gunshot location in national parks: a pareto front approach. *Sensors*, 9(12):9493–9512, 2009.

[GCnCMBRGP08] F.J. González-Castaño, E. Costa-Montenegro, J.C. Burguillo-Rial, and U. García-Palomares. Outdoor WLAN planning via non-monotone derivative-free optimization: algorithm adaptation and case study. *Computational Optimization and Applications*, 40(3):405–419, 2008.

[GD08] A. Ghosh and S.K. Das. Coverage and connectivity issues in wireless sensor networks: A survey. *Pervasive and Mobile Computing*, 4(3):303–334, 2008.

[GE01] L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 3, pages 1312–1320. IEEE, 2001.

[GFJ+09] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14. ACM, 2009.

[Gil61] E.N. Gilbert. Random plane networks. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):533–543, 1961.

[GKS03] S. Ganeriwal, R. Kumar, and M.B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149. ACM, 2003.

[GLGJ09] H. Ghasemzadeh, V. Loseu, E. Guenterberg, and R. Jafari. Sport training using body sensor networks: A statistical approach to measure wrist rotation for golf swing. In *Proceedings of the Fourth International Conference on Body Area Networks*, page 2. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[GN06] E. Gaura and R. Newman. Wireless sensor networks: The quest for planetary field sensing. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 596–603, 2006.

[GPBRGCn08] U.M. García-Palomares, J.C. Burguillo-Rial, and F.J. González-Castaño. Explicit gradient information in multiobjective optimization. *Operations research letters*, 36(6):722–725, 2008.

[GPGCnBR06] U.M. García-Palomares, F.J. González-Castaño, and J.C. Burguillo-Rial. A combined global & local search (CGLS) approach to global optimization. *Journal of Global Optimization*, 34(3):409–426, 2006.

[GPR02] U.M. García-Palomares and J.F. Rodríguez. New sequential and parallel derivative-free algorithms for unconstrained minimization. *SIAM Journal on Optimization*, 13:79, 2002.

[Gra06] S. Grant. Fertilizer efficiency for winegrape vineyards. *Practical Winery/Vineyard*, 28:35, 2006.

[Gua] Guardia Civil, Spanish Ministry of Interior. Servicio de Protección a la Naturaleza (SEPRONA). http://www.seproneros.com/index.php.

[Hai06] S.K. Hailes. The SEnsing for Sport And Managed Exercise project (SESAME), 2006. http://www.sesame.ucl.ac.uk.

[HB04] Y. Huang and J. Benesty. *Audio signal processing for next-generation multimedia communication systems*. Springer, 2004.

[HkS+09] M.A. Hussain, K. kyung Sup, et al. WSN research activities for military application. In *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, volume 1, pages 271–274. IEEE, 2009.

[HL+07] W.L. Haskell, I. Lee, et al. Physical activity and public health. updated recommendation for adults from the american college of sports medicine and the american heart association. *Circulation*, 2007.

[HLS+07] Y.F. Huang, W.H. Luo, J. Sum, L.H. Chang, C.W. Chang, and R.C. Chen. Lifetime performance of an energy efficient clustering algorithm for cluster-based wireless sensor networks. In *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, pages 455–464. Springer, 2007.

[HMP03] H.V. Huikuri, T.H. Makikallio, and J Perkiomaki. Measurement of heart rate variability by methods based on nonlinear dynamics. *Journal of Electrocardiology*, 36:95–99, 2003.

[HSA84] G.E. Hinton, T.J. Sejnowski, and D.H. Ackley. Boltzmann machines: Constraint satisfaction networks that learn. *Cognitive Science*, 9:147–169, 1984.

[HT05] C.F. Huang and Y.C. Tseng. The coverage problem in a wireless sensor network. *Mobile Networks and Applications*, 10(4):519–528, 2005.

[IA04] M. Ishizuka and M. Aida. Performance study of node placement in sensor networks. In *Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference on*, pages 598–603. IEEE, 2004.

[IKD04] V. Isler, S. Kannan, and K. Daniilidis. Sampling based sensor-network deployment. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 2, pages 1780–1785. IEEE, 2004.

[JB03] E.T. Jaynes and G.L. Bretthorst. *Probability theory: the logic of science*. Cambridge University Press, 2003.

[JTN+] T. Jaitner, M. Trapp, D. Niebuhr, J. Koch, and T.U. Kaiserslautern. Indoor-simulation of team training in cycling. *The Engineering of Sport 6*, page 103.

[KFP+91] D.T. Kaplan, M.I. Furman, S.M. Pincus, S.M. Ryan, L.A. Lipsitz, and A.L. Goldberger. Aging and the complexity of cardiovascular dynamics. *Biophysical Journal*, 59(4):945–949, 1991.

[Kri05] B. Krishnamachari. *Networking wireless sensors*. Cambridge University Press, 2005.

[KSG08] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9:235–284, 2008.

[KU02] M. Kamenetsky and M. Unbehaun. Coverage planning for outdoor wireless lan systems. In *Broadband Communications, 2002. Access, Transmission, Networking. 2002 International Zurich Seminar on*, pages 49–1. IEEE, 2002.

[KV88] J. Karvonen and T. Vuorimaa. Heart rate and exercise intensity during sports activities. practical application. *Sports Medicine (Auckland, NZ)*, 5(5):303, 1988.

[KW07] H. Karl and A. Willig. *Protocols and architectures for wireless sensor networks*. Wiley-Interscience, 2007.

[Lau57] A.G. Laurent. Bombing problems-a statistical approach. *Operations Research*, pages 75–89, 1957.

[LdSA89] M.C. López de Silanes and R. Arcangéli. Estimations de l'erreur d'approximation par splines d'interpolation et d'ajustement d'ordre (m, s). *Numerische Mathematik*, 56(5):449–467, 1989. d.

[LdSPPT01] M.C. López de Silanes, M.C. Parra, M. Pasadas, and J.J. Torrens. Spline approximation of discontinuous multivariate functions from scatterred data. *Journal of Computational and Applied Mathematics*, 131:281–298, 2001.

[LDWS08] B. Liu, O. Dousse, J. Wang, and A. Saipulla. Strong barrier coverage of wireless sensor networks. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '08, pages 411–420, New York, NY, USA, 2008. ACM.

[LJTL08] A. Le, T. Jaitner, F. Tobias, and L. Litz. A dynamic heart rate prediction model for training optimization in cycling(p 83). *The Engineering of Sport 7*, 1:425–433, 2008.

[LLJ08] A. Le, L. Litz, and T. Jaitner. A model predictive controller for sensor-based training optimization of a cyclist group. *The Engineering of Sport 7*, 1:413–423, 2008.

[LLL98] K. Lieska, E. Laitinen, and J. Lahteenmaki. Radio coverage optimization with genetic algorithms. In *Personal, Indoor and Mobile Radio Communications, 1998. The Ninth IEEE International Symposium on*, volume 1, pages 318–322. IEEE, 1998.

[LMVA09] P. López-Matencio and J. Vales-Alonso. IntPak. Ajuste e interpolación multivariable de funciones. In *3 CM, 3rd Congreso de Mathematica*. Salamanca, 2009.

[LMVA12a] P. López-Matencio and J. Vales-Alonso. ACO based approach to optimal node placement. In *ECUMICT- European Conference on the Use of Modern Information and Communication Technologies*, March 2012.

[LMVA12b] P. López-Matencio and J. Vales-Alonso. ACO based approach to optimal node placement. *[Under review on] AECE-Advances in Electrical and Computer Engineering*, April 2012.

[LMVAGCn+10] P. López-Matencio, J. Vales-Alonso, F.J. González-Castaño, J.L. Sieiro, and J.J. Alcaraz. Ambient intelligence assistant for running sports based on k-NN classifiers. In *Human System Interactions (HSI), 2010 3rd Conference on*, pages 605 –611, may 2010.

[Loc00] M. Locatelli. Simulated annealing algorithms for continuous global optimization: convergence conditions. *Journal of Optimization Theory and applications*, 104(1):121–133, 2000.

[LRS09] M. Leoncini, G. Resta, and P. Santi. Partially controlled deployment strategies for wireless sensors. *Ad Hoc Networks*, 7(1):1–23, 2009.

[LS86] P. Lancaster and K. Salkauskas. Curve and surface fitting. An introduction. *London: Academic Press*, 1, 1986.

[LT04] B. Liu and D. Towsley. A study of the coverage of large-scale sensor networks. In *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on*, pages 475 – 483, oct. 2004.

[LWS04] R. Lin, Z. Wang, and Y. Sun. Wireless sensor networks solutions for real time monitoring of nuclear power plant. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, volume 4, pages 3663–3667. IEEE, 2004.

[MFA07] G. Mao, B. Fidan, and B.D.O. Anderson. Wireless sensor network localization techniques. *Computer Networks*, 51(10):2529–2553, 2007.

[MKPS05] S. Megerian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava. Worst and best-case coverage in sensor networks. *Mobile Computing, IEEE Transactions on*, 4(1):84 – 92, jan.-feb. 2005.

[MRS08] C.D. Manning, P. Raghavan, and H. Schutze. *Introduction to information retrieval*, volume 1. Cambridge University Press, 2008.

[MS05] F. Michahelles and B. Schiele. Sensing and monitoring professional skiers. *Pervasive Computing, IEEE*, 4(3):40–45, Jul 2005.

[New] News HOY.es. Desarticulada un red de cazadores furtivos en la sierra de gredos. http://www.hoy.es/20090325/local/desarticulada-cazadores-furtivos-sierra-200903251611.html.

[NN03] D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1734–1743. IEEE, 2003.

[Non11] Nonin Medical Solutions. Self-Contained Pulse Oximeter and Sensor, 2011. http://www.nonin.com/OEMSolutions/iPod.

[Pat05] N. Patwari. *Location estimation in sensor networks*. PhD thesis, The University of Michigan, 2005.

[PCH+05] J. Pan, L. Cai, Y.T. Hou, Y. Shi, and S.X. Shen. Optimal base-station locations in two-tiered wireless sensor networks. *Mobile Computing, IEEE Transactions on*, 4(5):458–473, 2005.

[PHC04] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 95–107. ACM, 2004.

[PLB+06] D. Pfisterer, M. Lipphardt, C. Buschmann, H. Hellbrueck, S. Fischer, and J.H. Sauselin. MarathonNet: Adding value to large scale sport events-a connectivity analysis. In *Proceedings of the First International Conference on Integrated Internet Ad Hoc and Sensor Networks*, page 12. ACM, 2006.

[PLSS08] Y. Peng, R. Lahusen, B. Shirazi, and W.Z. Song. Design of smart sensing component for volcano monitoring. In *Intelligent Environments, 2008 IET 4th International Conference on*, pages 1–7. IET, 2008.

[Pol11] Polar. Team sports products from polar company, 2011. http://www.polarusa.com/us-en/b2b_products/team_sports/polar_team2_pro.

[PP03] M. Penrose and Oxford University Press. *Random geometric graphs*, volume 5. Oxford University Press Oxford, 2003.

[PPKS06] S. Poduri, S. Pattem, B. Krishnamachari, and G.S. Sukhatme. Sensor network configuration and the curse of dimensionality. In *Proc. Third Workshop on Embedded Networked Sensors (EmNets 2006), Cambridge, MA, USA*, 2006.

[PS04] S. Poduri and G.S. Sukhatme. Constrained coverage for mobile sensor networks. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 165–171. IEEE, 2004.

[PSLB08] P. Pérez Soriano and S. Llana Belloch. Instrumentation in sports biomechanics. *Journal of Human Sport and Exercise-University of Alicante*, 2(2):26–41, 2008.

[PSM+04] J. Polastre, R. Szewczyk, A. Mainwaring, D. Culler, and J. Anderson. Analysis of wireless sensor networks for habitat monitoring. *Wireless Sensor Networks*, 47(6):399–423, 2004.

[QMAHTG+09] A.F. Quiceno-Manrique, J.B. Alonso-Hernandez, C.M. Travieso-Gonzalez, M.A. Ferrer-Ballester, and G. Castellanos-Dominguez. Detection of obstructive sleep apnea in ECG recordings using time-frequency distributions and dynamic features. pages 5559–5562, Sep 2009.

[RLA06] J.L. Rouas, J. Louradour, and S. Ambellouis. Audio events detection in public transport vehicle. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 733–738. IEEE, 2006.

[Sad11] F. Sadri. Ambient intelligence: A survey. *ACM Computing Surveys (CSUR)*, 43(4):36, 2011.

[Sau] Kari Sauramo. Finnish suppressor project. http://www.guns.connect.fi/rs/measure.html.

[SB08] D. Spelmezan and J. Borchers. Real-time snowboard training system. In *CHI'08 extended abstracts on Human Factors in Computing Systems*, pages 3327–3332. ACM, 2008.

[SD08] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173, 2008.

[SH87] H. Szu and R. Hartley. Fast simulated annealing. *Physics Letters A*, 122(3 - 4):157 – 162, June 1987.

[Sil] Silvano Sistemas Sónicos S.L. Sistemas de vigilancia electrónica para fincas y cotos de caza. http://www.furtivos.com/index.php.

[SLF+08] T. Saponas, J. Lester, J. Froehlich, J. Fogarty, and J. Landay. iLearn on the iPhone: Real-time human activity classification on commodity mobile phones. *University of Washington CSE Tech Report UW-CSE-08-04-02*, 2008.

[Spa] Spanish Goverment. Spanish National Parks - Cabañeros. http://reddeparquesnacionales.mma.es/en/parques/cabaneros/index.htm.

[SPI11] Mars exploration rover mission: Home, October 2011. http://marsrover.nasa.gov/home/index.html.

[SPR96] H.D. Sherali, C.M. Pendyala, and T.S. Rappaport. Optimal location of transmitters for micro-cellular radio communication system design. *Selected Areas in Communications, IEEE Journal on*, 14(4):662–673, 1996.

[Ste] Stella Doradus. Antenna for MTS400 board. http://www.stelladoradus.com/2.4.ghz.planar.antennas.php.

[TCRB12] R. Thul, S. Coombes, H.L. Roderick, and M.D. Bootman. Subcellular calcium dynamics in a whole-cell model of an atrial myocyte. *Proceedings of the National Academy of Sciences*, 109(6):2150–2155, 2012.

[Tex] Texas Instruments (Chipcon product). CC2420 Chip Documentation. http://www.ti.com/lit/ds/symlink/cc2420.pdf.

[TIH+07] E.M. Tapia, S.S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman. Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor. In *Wearable Computers, 2007 11th IEEE International Symposium on*, pages 37 –40, oct. 2007.

[TMS01] H. Tanaka, K.D. Monahan, and D.R. Seals. Age-predicted maximal heart rate revisited. *Journal of the American College of Cardiology*, 37(1):153, 2001.

[UK03] M. Unbehaun and M. Kamenetsky. On the deployment of picocellular wireless infrastructure. *Wireless Communications, IEEE*, 10(6):70–80, 2003.

[Uns99] M. Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38, 1999.

[VACRBD+08] J. Vales-Alonso, S. Costas-Rodríguez, M. Bueno-Delgado, E. Egea-López, F. Gil-Castineira, P. Rodríguez-Hernández, J. García-Haro, and F. González-Castaño. An analytical approach to the optimal deployment of wireless sensor networks. *Computational Intelligence for Remote Sensing*, pages 145–161, 2008.

[VAELMS+07] J. Vales-Alonso, E. Egea-López, A. Martínez-Sala, P. Pavón-Mariño, M. Victoria Bueno-Delgado, and J. García-Haro. Performance evaluation of MAC transmission power control in wireless sensor networks. *Computer Networks*, 51(6):1483–1498, 2007.

[VALMA+12] J. Vales-Alonso, P. López-Matencio, J. Alcaraz, J. Sieiro-Lomba, E. Costa-Montenegro, and F. González-Castaño. A dynamic programming approach for ambient intelligence platforms in running sports based on markov decision processes. *Human–Computer Systems Interaction: Backgrounds and Applications 2*, pages 165–181, 2012.

[VALMAGH11] J. Vales-Alonso, P. López-Matencio, J.J. Alcaraz, and J. García-Haro. Decision support in AmI sport environments. In *IEEE Sensors conference*, 2011.

[VALMGC+10] J. Vales-Alonso, P. López-Matencio, F.J. Gonzalez-Castaño, H. Navarro-Hellín, P.J. Baños-Guirao, F.J. Pérez-Martínez, R.P. Martínez-Álvarez, D. González-Jiménez, F. Gil-Castiñeira, and R. Duro-Fernández. Ambient intelligence systems for personalized sport training. *Sensors*, 10(3):2359–2385, 2010.

[VAPGLMA12] J. Vales-Alonso, F.J. Parrado-García, P. López-Matencio, and J.J. Alcaraz. On the optimal random deployment of wireless sensor networks in non-homogeneous scenarios. *[Under review on] Ad Hoc Networks*, 2012.

[VCC99] K. Veropoulos, N. Cristianini, and C. Campbell. The application of support vector machines to medical decision support: a case study. *Advanced Course in Artificial Intelligence (ACAI'99)*, 1999.

[VGT+07] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti. Scream and gunshot detection and localization for audio-surveillance systems. In *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pages 21–26. IEEE, 2007.

[WXA08] D. Wang, B. Xie, and D.P. Agrawal. Coverage and lifetime optimization of wireless sensor networks with gaussian distribution. *IEEE Transactions on Mobile Computing*, pages 1444–1458, 2008.

[WXTH06] Q. Wang, K. Xu, G. Takahara, and H. Hassanein. Wsn04-1: Deployment for information oriented sensing coverage in wireless sensor networks. In *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*, pages 1–5. IEEE, 2006.

[XHTW10] K. Xu, H. Hassanein, G. Takahara, and Q. Wang. Relay node deployment strategies in heterogeneous wireless sensor networks. *Mobile Computing, IEEE Transactions on*, 9(2):145–159, 2010.

[YA08] M. Younis and K. Akkaya. Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Networks*, 6(4):621 – 655, 2008.

[YJ08] M. Yuchi and J. Jo. Heart rate prediction based on physical activity using feedforwad neural network. In *Convergence and Hybrid Information Technology, 2008. ICHIT'08. International Conference on*, pages 344–350. IEEE, 2008.

[ZBT+10] C. Zhang, X. Bai, J. Teng, D. Xuan, and W. Jia. Constructing low-connectivity and full-coverage three dimensional sensor networks. *Selected Areas in Communications, IEEE Journal on*, 28(7):984–993, 2010.

[ZC03] Y. Zou and K. Chakrabarty. Uncertainty-aware sensor deployment algorithms for surveillance applications. *GLOBECOM*, 5:2972–2976, 2003.

[zig] Zigbee Alliance Whitepapers. http://www.zigbee.org/LearnMore/WhitePapers.aspx.

[ZLL08] J. Zhang, J. Luo, and X. Luo. A robust localization algorithm for wireless sensor networks. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on*, pages 1–4. IEEE, 2008.